QL Today

Volume 16
Issue 3
March - May
2012

ISSN 1432-5454

The Magazine about QL, QDOS, Sinclair Computers, SMSQ...

The Story of the SER-USB Drivers
LCD Displays connected to the QL
Random Numbers
Raspberry Pi
and much more!

42 pages

www.QLToday.com

Contents

Advertisers
in alphabetical order

QL Today

ISSN 1432-5454

German office & Publisher:

Jochen Merz Software
Kaiser-Wilhelm-Str. 302
47169 Duisburg
Germany

Tel. +49 203 502011
Fax +49 203 502012
email: smsq@j-m-s.com
email: QLToday@j-m-s.com

Editor:

Geoff Wicks
Flat 5b
Wordsworth Avenue
Derby DE24 9HQ
United Kingdom

Tel. +44 1332 271366
email: glwicks@btinternet.com
email: QLToday@j-m-s.com

Co-Editor & UK Office:

Bruce Nicholls
38 Derham Gardens
Upminster
Essex RM14 3HA
United Kingdom

Tel +44 20 71930539
Fax +44 870 0568755
email: qltoday@q-v-d.demon.co.uk
email: QLToday@j-m-s.com

QL *Today* is published four times a year, our volume begins on beginning of June. Please contact the German or English office for current subscription rates or visit our homepage www.QLTODAY.com.

We welcome your comments, suggestions and articles. YOU make **QL *Today*** possible. We are constantly changing and adjusting to meet your needs and requirements. Articles for publication should be on a 3.5" disk (DD or HD) or sent via Email. We prefer ASCII, Quill or text87 format. Pictures may be in _SCR format, we can also handle GIF or TIF or JPG. To enhance your article you may wish to include Saved Screen dumps. PLEASE send a hardcopy of all screens to be included. Don't forget to specify where in the text you would like the screen placed.

If you need more information about the UNZIP program which is used by our BOOT program to unpack the files, we suggest that you visit Dilwyn Jones' web site where you find more information about lots of interesting QDOS software and INFOZIP at http://www.dilwyn.uk6.net/arch/index.html

The deadline for the next issue is the 6th of May 2012!

Editorial

by Geoff Wicks

Something very strange happened in the middle of January. We cannot remember if or when it has ever happened before, but it heralds good news for QL Today readers.

About three weeks before the copy date for this issue we had almost enough material to fill the magazine. To our surprise articles continued to come in until just after the deadline.

This has been a successful volume of QL Today. We have had a small rise in readership. The DVD that came with issue 1 was much appreciated and we received compliments for the content of issue 2. Issue 3 has 42 pages instead of the promised 32, and we have a small amount of material over for issue 4.

As we near the end of each volume we have to think about the future of QL Today and decide whether or not to continue. In 2010 and 2011 this decision was not easy. This year we have no doubts. There have been many positive developments. There will be a volume 17.

One welcome development has been the wider variety of topics. At one time there was little coverage of hardware, but it now has a firm place in the magazine. In this issue Adrian Ives graphically describes the frustrations of developing new products.

When Steve Poole could no longer write for us, we were worried about the coverage of SuperBasic. Last issue we looked at the SCALE command and in this issue Tobias Fröschle writes about RND and RANDOMISE. I have plans to cover another keyword in a future issue, but I wonder if you also have one you would like to write about.

One weakness in both QL Today and the Quanta Magazine has been games coverage. Peter Scott has written reviews for both magazines, but, given the amount of effort that RWAP Software has put into preserving and re-releasing games software, I would welcome more games content.

Recently the British education minister has been critical of computer education in schools, describing it as being boring lessons on using Word and Excel. He wants pupils to learn programming through writing games.

At about the same time I was surprised, and somewhat flattered, to discover four articles from the help page of my website on an e-book download site. One was my guide to user friendly programming which is heavily based on articles I wrote 13 years ago. Although it was written for the QL much of the content would be relevant to programmers for other systems. Perhaps even for today's school students learning programming.

Maybe we retro enthusiasts are not quite the dinosaurs some think us to be, but could become today's trend setters. Certainly the articles we have received this year indicate QL-ers have much still to offer.

In my final editorial for volume 15 I expressed a concern about a fall in the number of contributers and pleaded for more writers. Thanks to all of you who have responded. It is down to you that there will be a volume 17. Just keep those articles coming in!

Corrections

A number of errors crept into the last issue of QL Today.

References to the 2009 Quanta AGM in the last minute news on page 44 should have been to the 2010 Quanta AGM.

Norman Dunbar has also pointed out two errors in his Part 29 assembler article.

'Page 29: There seems to be a spare ".10" at the very end of the last line. The value for the number of IW Objects should be 0 and not 0.10.

Page 31: The final sentence in the paragraph above the heading "3. The Generated Code" should end after AppMenuTest1Win_asm. The chunk "The file should look similar..." appears to be a duplicate of the first sentence of the following section.'

Microdrive Slot Card Reader Nearer Reality

An SD card reader for the microdrive slot on native hardware has come a step closer after **Peter Graf** handed over the final stages of the project to **Memory Lane Computing**.

Early last year Peter Graf raised the possibility of an SD/MMC card reader for the QL and suggested several different possibilities, including the use of the QL ROM port, the parallel port on the Super Gold Card, the CPU socket or the microdrive slot. Most users expressed a preference for the last of these.

Late last year Peter released details of the final design:

"The QL-SD hardware system has two parts:

1) *A board which plugs into an internal QL ROM socket. It has a Minerva style EPROM socket on top. EPROMs with QL operating system plus QL-SD driver could be used here.*
2) *A board which is mounted behind a microdrive slot inside the original QL case. After the microdrive has been removed (which is quite simple) the new SDHC-card drive can be mounted with two screws. SDHC cards can be inserted just like microdrives :-) The drive LED for the SDHC card is separate from the original microdrive LED, so no internal wiring must be changed. It is visible beside the card socket.*

Both parts are connected by a simple ribbon cable."

Peter reports that the project has taken up much time that is no longer available and believes the best chance of making progress would be to transfer the task to Memory Lane Computing who have a good track record in producing innovative hardware and drivers. He has passed on all his schematics, parts lists, files for PCB manufacture and source code together with a small number of PCB prototypes. Peter adds that Memory Lane computing is entitled to release the product or modify it.

Head of Memory Lane Computing, **Adrian Ives** says of Peter's work:

"I would like to pay tribute to Peter's design and engineering skills in creating the QL-SD and getting the original C driver up and running. The product has always been his brainchild but we are very happy to be able to take it forward and will hopefully do that design justice."

He also summarised the present state of development:

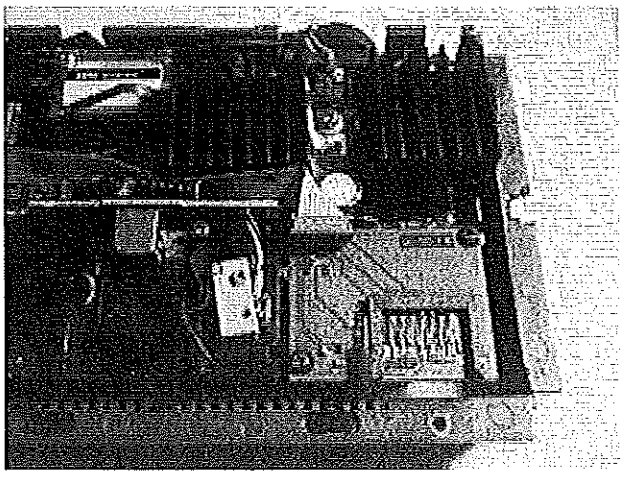
"The hardware is working on black box QLs but there are still some issues to be overcome relating to noise pickup. This is especially problematic if the SD card holder is mounted in a vacated microdrive slot. Peter has already redesigned the PCB to get around this problem.

There are problems with the Super Gold Card that have not yet been resolved. The Gold Card has not been tested at all (because I don't have one).

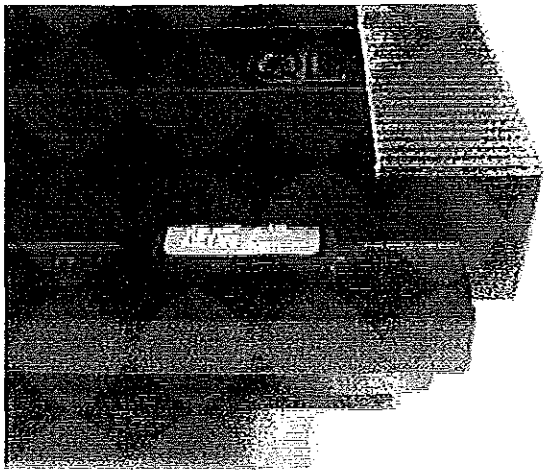
The software is written. There are two drivers available:

1) *QL-SD, a driver written in C derived from Dirk Steinkopf's QL-HD driver. At present this is too big to fit into a 16K EPROM and needs further optimisation.*
2) *An EDDE 2 driver that is both compatible with the Ser-USB and USBWiz over Q-BUS and is able to mount FAT32 volumes holding file system images. This is ROM-able and supports booting from an SD Card. All v2.x Ser-USB/Q-BUS utilities (Partition Manager etc) work with this and the partition formats are identical.*
3) *An EDDE 2 Block Device Interface driver for Q-emuLator's emulated hardware interface. This allows file system images to be accessed on the PC and read/written by QL software running under Q-emuLator.*

I currently have two functioning hardware prototypes:



QL SD - a view inside the open case



QL SD from outside

The QLROMEXT which plugs into one of the QL's internal ROM sockets. The SD Card carrier fits into a vacated microdrive position.
The QLROMEXTernal which plugs into the ROM port with the SD Card socket onboard."

The images of the device accompanying this report are taken from the QL modifications page of Lee Privett's website:
https://sites.google.com/site/theqlimagerepository/products
In other Memory Lane Computing news Adrian Ives has now discontinued the USB_SER product because the USBwiz module that it uses has reached the end of its life and is becoming difficult to obtain. It will also free his time to concentrate on other products. Support for existing users will be maintained.
Adrian hopes to attend the Manchester Quanta workshop at the end of March.

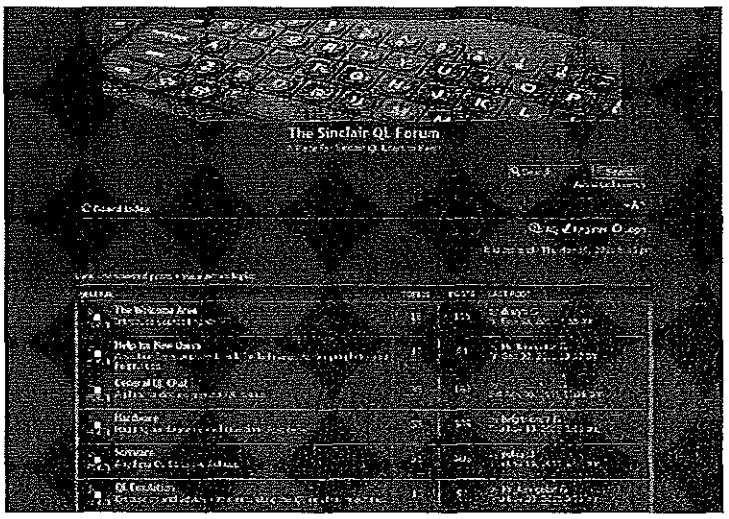
Crucial Votes at QUANTA AGM

Quanta members are to vote on the controversial six year rule and the terms of office of committee members at the AGM before voting on a new constitution. The draft constitution that was sent to members last summer retains the present arrangements, but feedback was invited from the members on whether they wanted changes.
Only one member provided feedback and the Quanta Committee feels the issues are of such importance that votes are necessary to clearly establish the desire of the members before final voting on the constitution.

The six year rule requires committee members to stand down for a least a year after completing six continuous year of office. The idea is to prevent the committee from becoming stale by encouraging fresh blood. Members will vote on whether or not to extend the rule from 6 years service to 9 years service.
Currently the present terms of office are 3 years for officers and 1 year for ordinary committee members. The proposal is for both officers and ordinary committee members to serve for 2 years.
Under the Quanta Constitution voting on constitutional matters requires a two-thirds majority for any changes to be made.

QL-FORUM Interview

The October/November issue of the Quanta Magazine contained an interview with the founders of the QL-Forum which celebrated its first anniversary at the end of November 2011.
The forum was set up by **Peter Scott** with **Rob Heaton** providing the technical help. Peter comments:
"We never thought about research, deciding just to build it and hope people will turn up. I knew it would be slow and steady progress initially just expecting a few posts a week with the odd person signing up but it's done far better than I expected."
The initial target was 100 members in the first year and 99 were achieved. There are about 2,000 page views a month and about 50% of the visitors are guests, some of whom will also be members who only sign on when they wish to comment. The forum has become international with posts from, among other countries, Italy, France, Germany, Holland and Sweden.
Mid February there were 117 members and 2101 posts on 323 topics.



Software Updates

George Gwilt has updated two programs:

SETW

SETW, which produces window definitions for TurboPTr, C and EasyPEasy has been updated to version 7.08. This allows the use of the largest possible names and directories. The names are used for the files produced by SETW. The directories can be set either by Config or by a parameter list. Directories can be set for such things as the output files. Thus if a directory is set for the assembler file the output will be put both in ram1_ and in the set directory.

GWASS

1. This allows the use of JSR.L adrs to mean JSR.L (adrs).L. The same applies to JMP. These work whether adrs is an explicit absolute address or a label.
2. Byte sized immediate ‹ea›s on assembly should now all produce a word with the first byte zero. Previously such an instruction as

   ```
   MOVE.B #-2,D0
   ```

   would have had $FFFE as the word. This will now assemble to $00FE.

http://gwiltprogs.info/

QEMULATOR

Daniele Terdina has announced a Q-emuLator blog, at

http://qemulator.blogspot.com/

He anticipates it will be a low traffic blog, mainly to inform existing users of new updates.

Updates for both Windows and OS X have just been released.

UK Postcodes

Just Words! has updated its postcodes program, jointly written with *Dilwyn Jones*, to use the new QL friendly map databases being developed. The program is available on the downloads page of the Just Words! website and now includes the source code. For any user wishing to experiment with the UK map, the mapping routines are in lines 10010 to 10210 and the database from 10230 to 12650.

www.gwicks.net/justwords.htm

The program can also be downloaded from Dilwyn's website:

www.dilwyn.me.uk/misc/index.html

News from Dilwyn Jones

PCB DESIGN V7.23

Malcolm Lear has released another update to his PCB Design program, which is available to download free at

http://www.dilwyn.me.uk/graphics/index.html

Malcolm says that there are *"a lot of changes this time, mostly removal of the dot matrix, HPGL and the obsolete Gerber-D drivers. This means a cleaner export program that's much easier to maintain. Gerber-X export has been updated by using the correct area fill commands rather than a plot fill which creates massive files. Component label size can now be adjusted by 5 mil steps rather than 20."*

eBooks Website

Several QL eBooks are now available to download from my website at

http://www.dilwyn.me.uk/docs/ebooks/index.html

This includes commonly used reference guides such as the QL User Guide, C68 manual and Toolkit 2 and Turbo manuals. I have prepared an article for QL Today on the subject of eBooks and why I have pursued this topic. The eBooks are available in ePub, Mobi and PDF formats, making them suitable for many eBook readers and tablet PCs. I am very grateful to people like Adrian Ives for the assistance given and in particular for converting many of the eBooks available so far.

QL Online Manual

Following on from the eBooks project, there is now an online copy of the QL manual, split into the four sections of

Introduction, Beginner's Guide, Keywords and Concepts sections. Some sections have an index, whereby you can click on a link in a list of keywords and be taken to the entry for that command or function in the Keywords Guide, for example. Hopefully now that this manual is available online, it will be easy to access anywhere where you have an internet connection, and will also be useful for new or returning QL users who may not have access to a paper manual.
The online QL manual is available at
http://www.dilwyn.me.uk/docs/ebooks/olqlug/index.htm

"THE SHELL" Update

Adrian Ives has released an updated version of the command shell for QDOS called The Shell.
Version 1.12 fixes a problem with matching nested pairs of brackets that incorrectly parsed expressions like $(%isfile:$(filename)).
Other changes include:
'set' is now a synonym of 'let'.
If an autoexec_bat startup command file is not found, the Shell will look for a file called autoshell_bat.
A special comment of the form #requires n.nn in a batch file will stop execution of the file if the Shell version number is less than that specified.
The Shell is available to download free from
http://www.dilwyn.me.uk/shells/index.html
Adrian has also revised some of the extensions in his package called extens.zip on the Toolkits page of my website – download it from
http://www.dilwyn.me.uk/tk/index.html

Q-STRIPPER Update

Norman Dunbar has released a recompiled version of his Q-Stripperprogram for Windows, which lets you view and convert Quill DOC files to other formats.
This version will now work as a 32-bit application on a Windows 7 64 bit system.
Remember that this is a Windows program, not a QL program!
Download Q-Stripper from
http://www.dilwyn.me.uk/filetran/index.html
If you need the compiler support files, these are a separate 5 megabyte download from the same page.

GST QL Software

Thanks to *Urs König* and encouragement from *Rich Mellor*, I have now added the GST QL software bundle to my website for all to download free on a non-profit basis. A while back, *Jeff Fenton* of GST gave permission for these programs to be downloaded on this basis.

68K/OS

I have created a new page on my website for the GST 68K/OS stuff at
http://www.dilwyn.me.uk/gst68kos/index.html ,
although it doesn't include the microdrive software part as I don't know how to convert it (it's not the same as QDOS microdrive format).
On this page you can run a 68K/OS system in a QemuLator demo which Urs had prepared.
If, like me, you get error messages when you try to use the demo QemuLator after unzipping everything, it is probably because you have another version of QemuLator already on your system - if so, use your registered version, go into the 'QL' menu, select 'QL Configuration' and find the GST_68K-OS BIN file to use as Main ROM in Qemulator and deselect any Back ROM such as a Toolkit 2 ROM image. Click OK. QemuLator should now start as a 68K/OS.

GST Q-MAC and QL Assembler

The Q-Mac and GST QL Assembler are both on the Assembler page on my website. The QL Assembler is under GST-ASM, and the Q-Mac is under Q-Mac on the page:
http://www.dilwyn.me.uk/asm/index.html

GST QC Development System

The GST QC development system is a C compiler system for the QL, originally supplied on microdrive cartridges, on my C page at
http://www.dilwyn.me.uk/c/index.html
under GST QC.

Various manuals, pictures and scanned adverts and reviews are also available as separate zip and PDF files on the pages. Some of them are quite large, e.g. the scanned QC manual is over 20MB, so take note of the file sizes before downloading (file sizes are in brackets alongside the descriptions).

J-M-S Email

My email address **SMSQ@j-m-s.com** has been heavily spammed for some months now (300... 1000 mails per day). This is the only account which gets so many junk mails, so I had to increase the spam filtering to 'much stronger'... impossible to look at so many junk mails manually. In case you write to me and get no reply, please use the freshly created **SMSQ@j-m-s.de** ... or better use this new email address in general in the future.

Quite Large Integers - Part 1

by George Gwilt

As I have already written, 64-bit arithmetic can be achieved by using C.

This was useful since the task of actually writing routines in assembler to do the arithmetical actions could be shelved. However, 64 bits is really quite small compared with many integers. I thus determined to find a way of dealing with larger numbers.

As it happened, one member of the Scottish QL Users Group (SQLUG) told me about an assembler program he had written to divide one large number by another. This was exciting. I would not need focus my thoughts on what I considered a difficult area. Another ray of light came when a second member of SQLUG suggested that it should be possible to apply the same method to multiplication as to division. I told him that this was nonsense. Later, of course, I realised that it was in fact eminently sensible.

In the programs explained below it is assumed that all numbers are positive.

First here is a general description of the methods used.

Addition and Subtraction

The Motorola 68000 chip has a very useful instruction which allows extended addition. This is:

```
ADDX.L    -(Ax),-(Ay)
```

Its purpose is to allow the extended number to the end of which Ax points to be added to the second extended number to the end of which Ay points. The instruction itself subtracts 4 from both Ax and Ay, adds together the long words to which the registers now point, adds the current value of the extend bit (X), sets a new value of the extend bit according to the result and stores the answer in Ay. As you can see, repeated application of this instruction will add together two extended integers. I call the routine to do this "addnm".

Subtraction can be done in exactly the same fashion using SUBX in a routine called "subnm".

Division

The method used to divide one number by another is really in essence very simple. You continually subtract a number from the dividend, each time reducing the dividend, until it becomes less than the divisor. Of course I did not just use the divisor as the number to be subtracted. What I did was to multiply the divisor by that power of 2 which made it just less than the dividend. This, of course, can greatly reduce the number of subtractions needed to get the answer.

However, there is the question of how to achieve multiplication by a power of 2. In essence this is simple enough. All you need to do is to shift up the bits of the multiplicand. It is easy enough to do this inside a register. for example:

```
LSL.L     #4,D0
```

will shift up the contents of D0.L four bits thus multiplying it by 2^4.

In an extended number this won't quite work. The shift can be expressed as a number of long words (N) with the residual number of bits (B). If B is zero the shift is just N long words. Otherwise we need to determine new values for the long words in the answer.

To find the new value of a long word I put it in D0 and use the instruction:

```
ROL.L     #4,D0
```

This shifts up the bits of D0.L as before, but any bits which go over the top reappear at the bottom of D0. These bits at the bottom of D0 will be those which need to be placed at the bottom of the next long word. We are counting backwards here, just as for ADDX. So, what we do is to roll round the bits in a long word, capture, separately, the bits at the bottom which have gone over the top and the bits at the top which have been shifted up. To these latter bits are added those bits which came over the top of the previous word. An iteration will give the shifted answer we need.

The routine performing this action is "shft".

One problem remains here, though. That is, how do we find how far to shift the divisor? We need to shift the divisor up so that its most significant bit is in line with that of the dividend, or perhaps is just one bit lower. As a first trial we align the numbers. To find the shift amount we need to know the position of the most significant bit in both integers. This, naturally,
requires yet another routine. This one I call "gbit", short for "get bit". By using this we get the positions of the top bit in both integers. By subtraction we find the amount of the shift.
As mentioned above, the shift can be expressed as N long words and B bits. In fact the position of the most significant bit in an integer is set by "gbit" as N in the upper word and B in the lower word of D1. If the positions in the two integers are N1|B1 and N2|B2 then the amount N1-N2|B1-B2, will give the correct amount to shift if both N1-N2 and B1-B2 are positive. If B2 › B1, then we will need to adjust the amount of the shift by adding 32 to the low word and subtracting 1 from the upper word.

Multiplication

At first sight multiplication is nothing like division. In division we keep subtracting from a known number until we get the answer. In multiplication of M by N we could get the answer by adding M to the answer N times. In the same way as for division, we make a short cut by shifting up the multiplicand the right number of bits for each bit in the multiplier and adding the shifted result. This is beginning to look like my method of division but with ADDX instead of SUBX.

To do this we need to know the positions of all the non-zero bits in the multiplier. To know when to stop the loop doing the shifting and adding we need to know also how many non-zero bits there are in the multiplier.

"gbit" comes to the rescue here, for it turns out that the same code can give us all these answers.

Program

It looks as though we can do the arithmetic on large integers using the six routines:

```
addnm    Add A to B
subnm    Subtract A from B
divs     Divide A by B
muls     Multiply A by B
gbit     Find number or position of bits
shft     Multiply number by power of 2
```

We actually need four more routines. These are required for housekeeping and testing. The additional routines are:

```
clrnm    Clears A
comp     Compares A with B
putbit   Sets a bit in the quotient during division
tstnm    Tests a number for positive, negative or zero
```

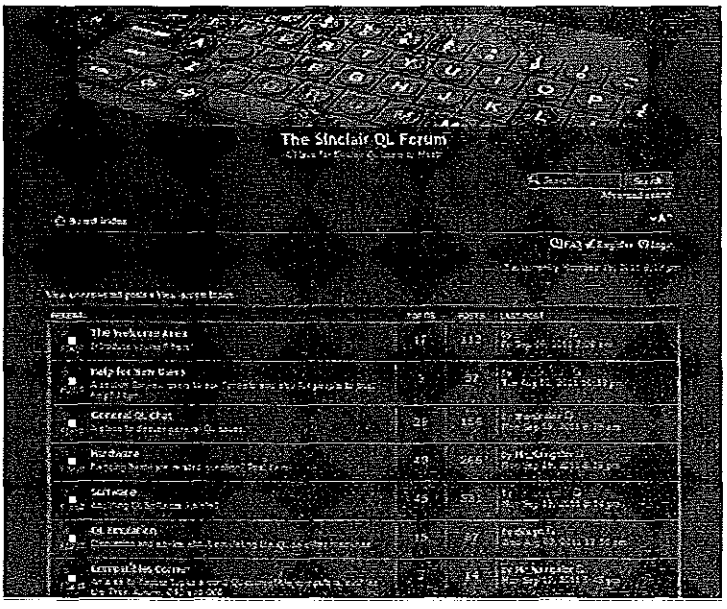
Before listing the routines it should be explained that each integer is stored in a certain number of long words. The number of these is contained in the word "sze". As an example this is taken here as 20. There are also two working spaces for numbers. These are called sp1 and sp2.

```
sze      dc.w      20        Number of long words holding integers

; addnm adds two numbers
; On entry : A0 and A1 point to the end of the numbers
; On exit  : D0=-1 if overflow, 0 otherwise

addnm:
         move.w    sze,d0              size of numbers . .
         subq.w    #1,d0               . . less 1 (clears the extend bit)
add1     addx.l    -(a0),-(a1)
         dbf       d0,add1
add3     bvs       add2                overflow (here D0,W = -1)
add4     moveq     #0,d0               good answer
add2     ext.l     d0                  sets D0.L 0 or -1
         rts
```

```
; subnm subtracts one number from another
; On entry : A0 and A1 point to the end of the numbers
: On exit  : D0 = -1 if overflow, 0 otherwise

subnm:
          move.w    sze,d0
          subq.w    #1,d0
sub1      subx.l    -(a0),-(a1)
          dbf       d0,sub1
          bra       add3

; divs divides z1 by z2 and sets the quotient to z3
; A0 -> z1
; A1 -> z2
; The quotient goes to z3
; and the remainder to z1
; A2 -> z3
; If z2 is 0, D0.L is set to -18

div_reg   reg       d5-7
divs      movem.l   div_reg,-(sp)
          movem.l   a0-1,-(sp)          keep pointers
          movea.l   a2,a0
          bsr       clrnm               clear answer space
          movea.l   a1,a0
          bsr       tstnm               is z2 zero? . .
          beq       divs5     ----->    . . yes -> OVR
          movea.l   (sp),a0
          bsr       tstnm               is z1 zero? . .
          beq       divs_end1           . . yes -> answer zero
          movea.l   sp2,a0
          bsr       clrnm               clear temporary answer
          move.w    sze,d0
          movea.l   a1,a0
          moveq     #-1,d1              get 1st bit position of z2 (p2)
          bsr       gbit
          move.l    d1,d7               p2

; The procedure loops here

divs4     movea.l   (sp),a0             A0 -> z1
          move.w    sze,d0
          moveq     #-1,d1              get 1st bit position of z1 (p1)
          bsr       gbit
          move.l    d1,d6               p1
          cmp.l     d6,d7
          beq       divs_same           p1 = p2
          bpl       divs_end1           p2 > p1 - finished

divs7
          move.l    d6,d0
          sub.l     d7,d0               amount of shift
          tst.w     d0
          bpl       divs1
          subi.l    #$ffe0,d0           adjust to positive mod
divs1     move.l    d0,d5               amount of shift to D5.L
divs6     move.w    sze,d1
          subq.w    #1,d1
          movea.l   sp2,a0
          bsr       clrnm               A0 set to end of sp2
          movea.l   a0,a1               A1 -> end of sp2
          movea.l   4(sp),a0            z2
          move.w    sze,d0
          lsl.w     #2,d0
          lea       (a0,d0.w),a0        to end of z2
          move.l    d5,d0               amount to shift
          bsr       shft                shifted value of z2 to sp2
```

http://qlforum.co.uk

A Place for Sinclair QL Users to Meet!



THE QL FORUM is an informal online community for talking about all things QL.

Join in and get involved!

```
          movea.l   (sp),a0             z1
          movea.l   sp2,a1
          bsr       comp                compare z1 and sp2
          beq       divs2               put answer and finish
          bhi       divs3               now shift one less and try again

; do subtraction here and set answer

          movea.l   sp2_e,a0            A0 -> end of sp2
          movea.l   (sp),a1             A1 -> z1
          move.w    sze,d0
          lsl.w     #2,d0
          lea       (a1,d0.w),a1        A1 -> end of z1
          bsr       subnm               z1 - sp2
          move.l    d5,d0               bit position
          bsr       putbit              set this position in the answer
          bra       divs4               do next bit

; The shift was one bit too much so try 1 less

divs3     tst.l     d5
          beq       divs_end1           finished
          subq.l    #1,d5               shift less 1
          tst.w     d5                  adjust shift? . .
          bpl       divs6               . . no
          subi.l    #$ffe0,d5
          bra       divs6

; dividend is now zero

divs2     movea.l   (sp),a0
          bsr       clrnm               clear remainder
          move.l    d5,d0
          bsr       putbit              set last bit in answer
divs_end1 moveq     #0,d0               OK end
divs_end  lea       8(sp),sp            reset stack for A0/1
          movem.l   (sp)+,div_reg
          rts

divs5     moveq     #-18,d0             OVR signalled
          bra       divs_end

; The bit positions are the same (no shifting needed)

divs_same movem.l   (sp),a0-1
          bsr       comp                compare z1 with z2 . .
          bne       divs7               . . not equal - continue loop
          moveq     #0,d5               bit position 0
          bra       divs2               dividend is now zero

; muls multiplies A by B and puts the answer to C
; On entry A0 -> start of A
;          A1 -> start of B
;          A2 -> start of C
; On exit  D0 is 0 if OK. CC are set

mul_reg   reg       d6-7
muls      movem.l   mul_reg,-(sp)
          movem.l   a0-1,-(sp)          keep pointers
          movea.l   a2,a0
          bsr       clrnm               clear answer space
          movea.l   a0,a2               point to end of answer

; Test numbers for zero

          movea.l   (sp),a0
          bsr       tstnm               test A
```

```
          beq       muls4               A = 0
          movea.l   4(sp),a0
          bsr       tstnm               test B
          beq       muls4               B = 0
          movea.l   (sp),a0             A0 -> A

; Test if shift would give OVR

          move.w    sze,d0
          moveq     #-1,d1
          bsr       gbit                find 1st bit of A
          move.l    d1,d7               1st bit position to D7.L
          movea.l   4(sp),a0            A0 -> B
          moveq     #-1,d1
          move.w    sze,d0
          bsr       gbit                find 1st bit of B
          add.l     d7,d1               add positions of A and B
          subq.w    #1,d1               one bit less
          lsr.w     #5,d1               longwords (+1, 0 or -1)
          move.l    d1,d7
          swap      d1
          add.w     d7,d1               estimate of no of long words
          cmp.w     sze,d1              too big? . .
          bgt.s     muls1     ----->    . . yes

; Set the number with least bits as multiplier

          movea.l   (sp),a0             A
          move.w    sze,d0
          moveq     #0,d1
          bsr       gbit                get no of bits in A . .
          move.w    d1,d7               . . to D7
          movea.l   4(sp),a0            B
          move.w    sze,d0
          moveq     #0,d1
          bsr       gbit                get no of bits of B . .
          move.w    d1,d6               . . to D6
          cmp.w     d6,d7
          ble       muls2               d7 < d6
          movem.l   (sp),a0-1
          exg       a0,a1               switch A and B
          movem.l   a0-1,(sp)
          move.w    d6,d7               A0 to least bits
muls2
          neg.w     d7                  number of bits in multiplier
muls3     movea.l   sp2,a0
          bsr       clrnm               clear sp2
          move.w    d7,d1
          move.w    sze,d0
          movea.l   (sp),a0
          bsr       gbit                D1 = position of rth bit
          movea.l   4(sp),a0            B to A0
          move.w    sze,d0
          lsl.w     #2,d0
          lea       (a0,d0.w),a0        -> end of number B
          movea.l   sp1,a1
          move.l    d1,d0
          bsr       shft                shift to sp1
          tst.w     d0
          bne       muls1     ----->    OVR
          movea.l   sp2_e,a0
          movea.l   a2,a1               Answer address end
          bsr       addnm               add shifted multiplicand
          bne       muls1     ----->    OVR
          addq.w    #1,d7               ended? . .
          bne       muls3               . . not yet
```

```
          movea.l   sp2,a0
          bsr       clrnm               clear sp2
          moveq     #0,d0               mark OK
muls4     lea       8(sp),sp            tidy stack
          movem.l   (sp)+,mul_reg
          rts

muls1     moveq     #-18,d0             OVR
          bra       muls4

; gbit counts the bits in a number or returns position
;
; On entry:
;  D0.W = number of long words in number
;  D1.W = 0 for count and -r for position of rth non zero bit
;  A0 -> the start of the number
;
; On exit:
;  D1.W = number of bits or D1.TOP = bit DIV 32 : D1.W = bit MOD 32
;  If D1.W is negative there is no bit for the position requested.
;
; No other registers are used

gbit_reg  reg       d2-3
gbit      movem.l   gbit_reg,-(sp)
gbit5     tst.w     d0
          beq       gbit1               no more words
          subq.w    #1,d0               count down
gbit2     tst.l     (a0)+               look for non-zero lwd
          dbne      d0,gbit2
          beq       gbit1               finished
          move.l    -4(a0),d3           1st non zero word
          moveq     #31,d2              bit count less 1

gbit3     lsl.l     #1,d3               search for non zero bit
          dbcs      d2,gbit3            branch till bit found
          addq.w    #1,d1               one bit found and . .
          beq       gbit4               . . its place found
          subq.w    #1,d2               adjust bit count since the . .
;                                       . . DBCS did not alter D2.W
          tst.l     d3                  any bits left here? . .
          beq       gbit5               . . no : go to next long word
          bra       gbit3               finish this lwd
gbit1     movem.l   (sp)+,gbit_reg
          rts

; This sets the position of the rth non-zero bit

gbit4     move.w    d0,d1               bit DIV 32
          swap      d1
          move.w    d2,d1               bit MOD 32
          bra       gbit1


; Shift up
;
; On entry D0.TOP = shift number div 32 - W
;          D0.W   = shift number mod 32 - B
;          A0 -> end of number
;          A1 -> end of answer space
; On exit  D0.W = 0 if no overflow
; No other registers are used

; S - W long words of A0 are shifted to long words of A1
; starting from the end at long word W. (S is size of numbers).
;
; Inside  D3 = long word operand
```

```
;         D4 = 1st storage of bottom bit
;         D5 = 2nd storage of bottom bit
;         D6 = bottom mask
;         D7 = top mask

; Note that an error occurs if:
;  1. The shifting sets non-zero bits too high up (D5 non zero).
;  2. There are non zero words at the top not to be shifted.
;  3. The answer is negative.

sh_reg    reg       d2-7

shft      move.w    sze,d1
          subq.w    #1,d1
          tst.l     d0                  if D0 = 0 . .
          beq       copy1               . . just copy
          movem.l   sh_reg,-(sp)
          move.l    d0,d2
          swap      d2                   W (n div 32)
          sub.w     d2,d1               adjust count
          lsl.w     #2,d2               prepare . .
          suba.w    d2,a1               . . answer address
          lsr.w     #2,d2               restore long word count
          moveq     #1,d6               prepare masks
          lsl.l     d0,d6
          subq.l    #1,d6
          move.l    d6,d7
          not.l     d7
          moveq     #0,d5               set previous bottom bit to 0

shft1     move.l    -(a0),d3            get long word
          rol.l     d0,d3               shift it
          move.l    d3,d4               keep . .
          and.l     d6,d4               . . bits shifted out
          and.l     d7,d3               clear the shifted bits
          or.l      d5,d3               insert previous shifted bits
          move.l    d3,-(a1)            insert altered long word
          move.l    d4,d5               prepare for next long word
          dbf       d1,shft1            count all long words of number
          bne       shft2               D5 not zero (error)
          bra       shft3
shft4     tst.l     -(a0)               look at remaining words in A0
shft3     dbne      d2,shft4
          beq       shft2               no error (and D5 = 0)
          moveq     #1,d5               signal error
shft2     move.w    d5,d0               signal any overflow
          bne       shft5               error already found
          tst.l     d3                  is this negative? . .
          bpl       shft5               . . no OK
          moveq     #1,d0               set error
shft5     movem.l   (sp)+,sh_reg
          rts

; clrnm clears the number A0 points to the start of

clrnm     move.w    sze,d0
          subq.w    #1,d0
clr1      clr.l     (a0)+
          dbf       d0,clr1
          rts


; comp tests two numbers for equality
; On entry A0 -> 1st and A1 -> 2nd
; D0, A0 and A1 used.
; On exit condition code Z is set for equality else not
```

```
comp      move.w    sze,d0
          subq.w    #1,d0
comp1     cmpm.l    (a0)+,(a1)+
          dbne      d0,comp1
          rts

; copy copies A0 to A1
; A0 -› end of 1st number
; A1 -› end of 2nd number

copy      move.w    sze,d1
          subq.w    #1,d1
copy1     move.l    -(a0),-(a1)
          dbf       d1,copy1
          moveq     #0,d0               signal no overflow
          rts

; putbit sets the bit in a number
; A2 points to the start of the number
; D0 contains the bit position p [ p div 32 | p mod 32 ]
; Uses D1 and A0 : D0 unaltered

putbit    move.w    sze,d1
          lsl.w     #2,d1
          lea       (a2,d1.w),a0        A0 -› end of number
          move.l    d0,d1
          swap      d1
          lsl.w     #2,d1               Number of lwds
          suba.w    d1,a0               Point to end of lwd to be altered
          move.l    -(a0),d1
          bset      d0,d1               Mark bit
          move.l    d1,(a0)             Set altered lwd
          rts

; tstnm test a number for neg, zero or positive
; con codes set mi, eq or gt
; if number is $80000 . . . mi set and D0 = -1
; If number is ‹0 and not $80000 . .  D0›0

tstnm     move.w    sze,d0              size . .
          subq.w    #2,d0               . . less 2
          tst.l     (a0)+               1st long word
          bmi       tst3                negative
          bne       tst1                non zero

; now test for zero

tst2      tst.l     (a0)+
          dbne      d0,tst2
          addq.w    #1,d0               D0 is zero if number is
tst1      rts

tst3      move.l    -4(a0),d0           1st long word again
          lsl.l     #1,d0               shift up 1
          bne       tst4                set d0›0 and neg CC
          move.w    sze,d0
          subq.w    #2,d0
tst5      tst.l     (a0)+
          dbne      d0,tst5
          bne       tst4                OK - not $80000000
          ext.l     d0                  sets cc neg and d0 -1
          rts

tst4      moveq     #1,d0               Signal $80000000
          move      #8,ccr
          rts
```

Some Random Thoughts on Random Numbers

by Tobias Fröschle

Recently, there was a discussion in the QL Forum about QL S*BASIC RND and RANDOMISE. I thought this topic would be a nice one for a QL publication.

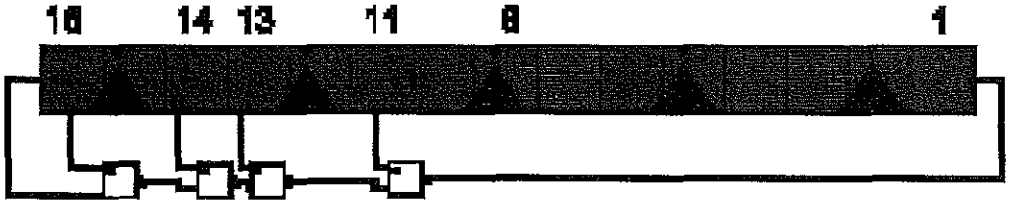
Creating random numbers is for a computer – built as a machine that handles exactly reproducible calculations – one of the hardest tasks to do. So anything a computer does regarding random numbers is, in a mathematical sense, considered *pseudo-random*. Computer scientists are content when a sequence of random numbers *looks random*. The following are some requirements to be met by a computer's random number generator (RNG):

Equal distribution of random numbers across the target range (the probability of getting a specific value should be the same for all the values in the target range)

The sequence of numbers delivered by the RND function should not be easily detectable.

It should be guaranteed that no specific value is excluded from the sequence of generated numbers (this aligns with (1) but is a bit different) – This means, that when run for a very long time, the sequence of generated random numbers should contain all possible values.

How is this achieved in SuperBASIC?
I have not checked in the real system, but in theory, the generation of pseudo-random numbers works as follows (and I would expect the QL's RNG doesn't work much differently):



When a new random number is needed, the register is shifted to the left and the bit shifted out XORed with specified bit positions in the register and shifted in from the right. The resulting value is then scaled into the range needed (RND (x TO y)) or interpreted as a floating point value between 0 and 1 (in the case of RND with no argument). This procedure generates a sequence of random numbers. The "quality" of a random number generator is measured by the requirements given above – the method shown in the picture (where the bit positions 16, 14, 13, and 11) are used by the feedback loop has proven especially good in that respect.
Another term used in pseudo random number generation is the so-called *seed*, which is nothing else but the initial value of the register being shifted around. This naturally affects the sequence of numbers being provided.
The QL system variable SV_RAND holds the system random number register, while the Basic Variable BV_RAND holds the same for the super basic job(s).

RANDOMISE

Simple – This just sets the initial value in the shift register. From there on, the RND function provides numbers starting from there. If you give an argument to RANDOMISE, you are guaranteed to get the same sequence of numbers in every run of your program.
If you don't give an argument, the S*BASIC RNG (BV_RAND) is initialized from the system RNG (SV_RAND). This makes it "truly" random (as SV_RAND seems to be initialized based on the system clock somehow – It is also used for the Microdrive random IDs).

RANDOMISE is probably the most misunderstood BASIC function (regardless on what computer). Most people seem to think this should be used to make RND *more random*. The opposite is the case. RANDOMISE is used to make the sequence of random numbers generated from RND *reproducible*.

Without using RANDOMISE, RND should be just about as random as possible (at least when you are not starting your program from a BOOT file which could result in the same timing between the start of the computer and the execution of your program)

The easiest way to make RND as random as possible is:

```
100 REMark Random
110 PRINT "Press any key to continue" : PAUSE -1 : RANDOMISE
```

This introduces the user waiting until a key is pressed as some sort of random seed.

Fancy Uses of the RNG

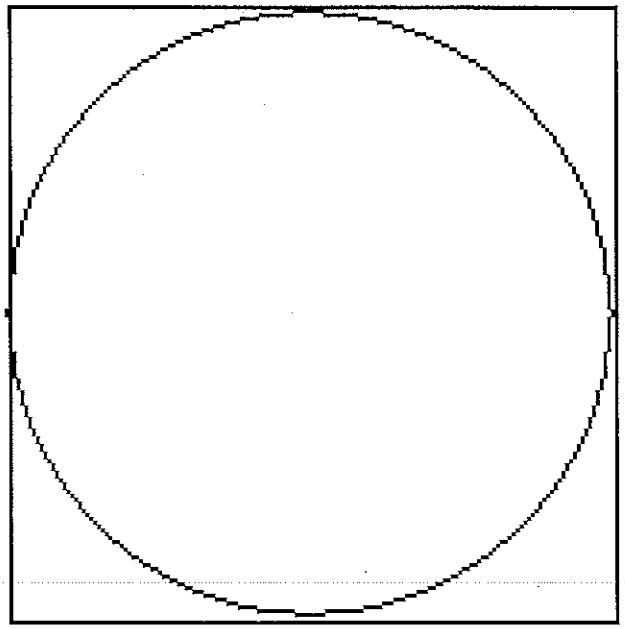
Apart from the more obvious uses for the RNG – like rolling dices or creating other random numbers, you can use random numbers (with the proper seed) as a source for reproducible sequences of numbers or as a source of statistical input. The sample programs have been created on QPC, but should equally run on any QL with at least TK2 fitted.

PI and Rainfall

This seems to be a method for calculating PI that has been invented in the British Islands (Wales, probably) or in some similar rainy landscape.

Ingredients:

1. An exact square of 2 x 2 meter (or foot, or whatever your favourite measure might be) of ground area
2. a circle of radius one centred within that square
3. rainfall
4. (lots of) patience



Let there be rain (random) on our square. Count the raindrops that fall into the square. Additionally, count the amount of raindrops that also fall into the circle. Calculate PI from the proportion of the two figures. In our program, lines 120 to 200 iterate over single raindrops. The "landing" position in our area is randomly generated in line 130, line 140 checks whether the raindrop landed in the circle or not.

The Circle area calculates as

```
c = PI r^2
```

The square is

```
s = 4 r^2
```

Eliminating r^2 and inserting the equations into each other, PI evaluates to

```
4c / s
```

This is the formula used in line 190 to calculate our current approximation of PI.

The check whether the raindrop falls into the circle uses good old Pythagoras to check whether the radius from the origin falls within the circle.

The program also shows the rain falling on our experiment graphically. I have used OVER -1 to be able to see rain falling even when the complete figure is already filled with "raindrops".

```
100 REMark CalcPI
110 Initialise
120 REPeat loop
130   x = RND : y = RND
140   IF isInCircle (x, y) THEN
150     in = in + 1
180   END IF
```

QUANTA

Independent QL Users Group

World-wide Membership is by subscription only,

Offering the following benefits: Bimonthly Magazine - up to 52 pages

Massive Software Library - All Free! : Free Helpline and Workshops

Regional Sub-Groups. One near you?

Advice on Software and Hardware problems

1 year Membership Subscription £18 (includes eMag)

If you want a printed copy of Quanta magazine, add the postage rates below

UK Postage £2.50 - Europe Postage £10 - International Postage £14

PayPal Surcharge about 5% - PayPal (see QUANTA Web Site)

Cash, Cheques and Postal Orders Accepted

*****Now in our Twenty Ninth Year*****

Further details from the Membership Secretary

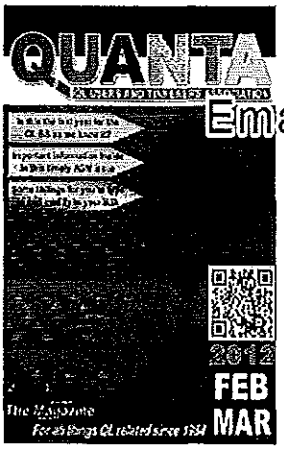
Keith Dunbar, 44, Dalton Avenue,

Stretford, Manchester, M32 9TP (UK).

Tel. 07789 483 800

Email: membership@quanta.org.uk

http://www.quanta.org.uk



Email: membership@quanta.org.uk and ask about our special 3 Year discount

```
182   out = out + 1
185   POINT #inner, x, y
190   AT #outer, 11,10: PRINT #outer, 4*in/out;"    "
200 END REPeat loop
210 :
220 DEFine PROCedure Initialise
230   outer = FOPEN ("con_")
240   inner = FOPEN ("con_")
250   WINDOW #outer, 138, 140, 18, 18
260   PAPER #outer,0
270   INK #outer,2
280   BORDER #outer,2,1
290   CLS #outer
300   WINDOW #inner, 135,100,20, 20
310   SCALE #inner, 2,-1,-1
320   INK #inner, 0
330   PAPER #inner, 4
340   CLS #inner
350   FILL #inner, 1 : CIRCLE #inner, 0, 0, 1 : FILL #inner,0
360   INK #inner, 2
365   OVER #inner,-1
370   in = 1
380   out = 1
390   AT #outer, 10, 0: PRINT#outer, "PI:       "; PI
400   AT #outer,11, 0: PRINT#outer, "Calc:     ";
410 END DEFine
420 :
430 DEFine FuNction isInCircle (x, y)
440   IF SQRT(x * x + y * y) < 1 THEN
450     RETurn 1
460   ELSE
470     RETurn 0
480   END IF
490 END DEFine
```

This is probably not the fastest method to calculate PI (On my QPC laptop it took more than an hour to converge to something reasonably resembling PI), but it's an interesting method anyway, as it both tests the quality of the RNG and the floating point precision of your computer.

How to hide your secrets using RND

Another interesting use of the Random Number Generator is ciphering of text. As the RNG generates reproducible sequences of random numbers when seeded with the proper values, it can be used to decode and encode secret messages. Funny thing is: Without a QL, nobody (well, except the NSA, probably) can decode your messages. I'm not even sure whether the RNG has changed through the lifetime of the various QL incarnations, i.e whether you would be able to decode a file ON QPC that has been created on a real QL.

The program uses the constant string in line 10005 as the string to cipher. It is encoded using the method described below and written to a file on ramdisk. The second part of the program (starting with line 10040) reads that encoded file, asks for the decoding key and translates it back to something readable, provided that you still remember the key (an arbitrary floating point number)..

```
10000 out = FOP_NEW("ram1_testfile")
10005 a$ = "Hello World - This is secret"
10010 z$ = encode$(a$)
10020 PRINT #out;z$
10030 CLOSE #out
10040 in = FOPEN ("ram1_testfile")
10050 INPUT #in, b$
10060 PRINT "Secret String:";b$
```

```
10070 c$ = encode$ (b$)
10080 PRINT "Decoded:";c$
10085 CLOSE #in
10090 :
10100 DEFine FuNction encode$(x$)
10105   LOCal i,a, b, c
10107   y$ =""
10110   INPUT "Key:";a
10120   RANDOMISE a
10130   FOR i = 1 TO LEN (x$)
10135     a = CODE(x$(i))
10137     b = RND (32 TO 128)
10138     c = a ^^ b
10140     y$=y$&CHR$(c)
10150   END FOR i
10155   PRINT
10160   RETurn y$
10170 END DEFine
```

The interesting part is in the function encode$. The RNG is initialised with the secret key value that needs to be the same for encoding and decoding. The individual characters of the string to encode are then XORed with the resulting sequence of random numbers from the RNG.

Programming in Assembler, Part 30 Creating and Using Libraries with GWASL

by Norman Dunbar

Introduction

At the end of the last issue, I promised to continue looking at Application Sub-Windows by adding some code to our menu enabled program. Unfortunately, due to the current very busy situation at work, and a mild dose of tendonitis in my thumb, I'm having to do a lot of one-handed typing these days which is slowing me down a lot. To this end, I'm taking a break from application menus for a wee while, and this issue, my article will be small – but hopefully, perfectly formed – looking at how we can create and use our own libraries of useful routines with gwasl.

The Library Code

The following is the complete code for a small library that allows your own assembly code to clear various parts of the screen. I apologise for the briefness of this article, but as I said, I'm typing one handed at the moment. The code should be typed into a file named lib_cls_asm or something similar.

```
;=======================================================;
; lib_cls_asm.                                          ;
;=======================================================;
; A small library to demonstrate the use of same in GWASL ;
; It's not particularly useful, it only demonstrates a    ;
; point!                                                  ;
;=======================================================;
; All routines expect the channel id in A0.L.           ;
; All routines assume infinite timeout.                 ;
; All regsiters are preserved, except D0.L.             ;
; Error codes are returned in D0.L and the Z flag.      ;
```

```
;=======================================================;

cls_screen equ $20
cls_top    equ $21
cls_bottom equ $22
cls_line   equ $23
cls_end    equ $24

infinity   equ -1

;-------------------------------------------------------;
; CLEAR_SCREEN - Clears entire screen.                  ;
;-------------------------------------------------------;
clear_screen  moveq #cls_screen,d0
              bra.s just_do_it

;-------------------------------------------------------;
; CLEAR_TOP - Clears top of screen.                     ;
;-------------------------------------------------------;
clear_top     moveq #cls_top,d0
              bra.s just_do_it

;-------------------------------------------------------;
; CLEAR_BOTTOM - Clears bottom of screen.               ;
;-------------------------------------------------------;
clear_bottom  moveq #cls_bottom,d0
              bra.s just_do_it

;-------------------------------------------------------;
; CLEAR_TO_EOL - Clear to end of cursor line.           ;
;-------------------------------------------------------;
clear_to_eol  moveq #cls_end,d0
              bra.s just_do_it

;-------------------------------------------------------;
; CLEAR_LINE - Clears entire cursor line.               ;
;-------------------------------------------------------;
clear_line    moveq #cls_line,d0

just_do_it    movem.l d1/d3/a1,-(a7)
              moveq #infinity,d3
              trap #3
              movem.l (a7)+,d1/d3/a1
              tst.l d0
              rts
```

So, you can see that there's not much to it.

That's the end of step one. The next step is to assemble the file using gwasl in the normal manner, fix any errors, and create an output file most likely named lib_cls_bin. In addition to the binary file, there will be another symbol file named lib_cls_sym created. We need that file shortly, however, it isn't in a format we can use just yet.

Once all errors have been removed and the source assembled, we are ready to move onto creating our library. In actual fact, half of the library is already created – lib_cls_bin – but we need to convert the symbol file into a text file that we can include in our own source code in order to actually call the routines in the library.

Execute the utility named sym_bin in your gwasl directory. The layout of the screen should look pretty familiar if you have used gwasl frequently. Choose option 1 as normal, and type in the path to the lib_cls_sym file.
After a couple of seconds, you can choose the option to exit. Our work is done!

Sym_bin has taken the binary formatted lib_cls_sym file and created from it a new text file named lib_cls_sym_lst. If you open this in an editor, it will look something like the following:

```
CLS_SCREEN      EQU     $00000020
CLS_TOP         EQU     $00000021
CLS_BOTTOM      EQU     $00000022
CLS_LINE        EQU     $00000023
CLS_END         EQU     $00000024
INFINITY        EQU     $FFFFFFFF

CLEAR_SCREEN    EQU     *+$00000000
JUST_DO_IT      EQU     *+$00000012
CLEAR_TOP       EQU     *+$00000004
CLEAR_BOTTOM    EQU     *+$00000008
CLEAR_TO_EOL    EQU     *+$0000000C
CLEAR_LINE      EQU     *+$00000010
```

You can see that all the equates defined in our source code have been made visible as well as offsets to the various routines. These offsets are the actual addresses within the lib_cls_bin file where the individual routines start.

It would be nice if there was some way for equates etc within a library to be invisible from outside it without us having to do too much extra work, however, as far as I'm aware, it's not possible to define an equate as "local" – similar to SuperBasic. What we can do is delete the top few lines leaving only the offsets to the routines. Edit the file to delete the lines from CLS_SCREEN down to INFINITY.

Next, create a new file containing these two lines, call it lib_cls_in:

```
in  win1_gwasl_libs_lib_cls_sym_lst
lib win1_gwasl_libs_lib_cls_bin
```

And that's all there is to it. To use the code simply include the following at the end of your own assembly code:

```
in  win1_gwasl_libs_lib_cls_in
```

Obviously, your paths will be different from mine, so change accordingly to suit your own system.

I have combined the IN and the LIB commands into one single file because I like to do as little typing as possible. You need not do this and to use the library, simply add the two lines above into the end of your own code at some point.

To demonstrate the code, all you need is something like this. Not shown in this example are other libraries that I use to set colours, open screens etc.

```
start           bsr     open_scr            ; Open a screen channel. Return id in A0
                bsr     set_colours         ; Set paper, strip and ink to my defaults
                                            ; Preserves all registers except D0.L
                bsr     clear_screen        ; Clear screen
                ...

                in  win1_gwasl_libs_lib_cls_in
                in  win1_gwasl_libs_lib_defaults_in
                in  win1_gwasl_libs_lib_colours_in
```

End Of Chapter 30

So, that's the end of this exciting instalment. Hopefully next time I'll be able to continue where I left off last time. I should also point out that George has created a new version of EasyPEasy with some changes to make setting up menus dynamically at run time much easier. I shall be looking at the changes next time.

Glossary of Abbreviations and Terms Part 2 - D to G

by Dilwyn Jones and Lee Privett

We continue here from where we ended two issues ago.

Database	An application that stores and manages data e.g. Archive or DB on the QL.
DB	Data Base, also the term used for the USA equivalent of Archive
DBF	Database file
DD	Double Density, normally refers to a type of floppy disk or drive
Debug	Examine a program to find out why it or some part of it isn't running as it ought to.
Debugger	A program which lets you run your program in a manner (usually line by line) which lets you examine values of variables etc to help you debug and work out why your program isn't running the way it ought to.
DIL	Dual In Line, normally used when referring to the type of IC sockets used on a circuit board
DIN	Deutsche Industrie-Norm, the German equivalent of BSI and ANSI, many types of audio and power connectors are often referred to as a 'DIN' plug or 'DIN' socket
DIMM	Dual Inline Memory Module, a type of memory card used by PCs using both sides of the card
DIYTK	Do It Yourself Toolkit. Name used to refer to a long running series of articles by Simon Goodwin in QL World magazine, where he wrote extensions software for the QL as individual files which you could bundle any of them together into a boot file, hence the DIY name
Dongle	Term used to refer to the plug in cartridge issued with the first QLs where part of the operating system was held in a small ROM cartridge plugged into the ROM expansion slot at the back of the QL. The term 'Kludge' was sometimes used as a synonym. Both terms were originally used by early reviewers of the QL. Outside the QL scene, this term has also been used to mean either a plug-in cartridge used like a key, without which the computer or a particular software will not work unless you have the key (or 'dongle'). More recently, the term 'dongle' has been used to refer to the plug in wireless networking devices for PCs, for example.
DS	Double Sided, normally refers to a type of floppy disk or drive
DTE	Data Terminal Equipment, RS232C communications term
DLL	Dynamic Link Library, an interface allowing a programmer to use code from within his/her own application
DOS	Disk Operating System
DPI	Dots per inch, used to describe print density on a printer, for example
DRAM	Dynamic Random Access Memory. The information stored in DRAM is lost if the power is turned off
Driver	Name given to a piece of software which allows the computer to control a specific type or piece of hardware connected to that computer.
DTP	Desk Top Publishing

ED	Extra Density or Extra-high Density, refers to the 3.2 megabyte floppy disks for the QL, or their disk drives
EE	Extended Environment, a term used to describe the combination of PTR_GEN, WMAN and HOT_REXT (or the equivalent in SMSQ/E) which give you a system which enhances your QL by saving and restoring window contents, hotkeys, standard menus and so on
EGA	Enhanced Graphics Adaptor for the PC, now largely superseded, this term is still used to refer to a particular type of screen display. On the QXL, for example, an EGA display mode refers to a 640x350 pixel display.
E-MAIL	Electronic Mail, commonly used by Internet enthusiasts to send messages etc. to each other via the Internet
EPROM	Erasable Programmable Read-Only Memory, a special memory chip, which can be programmed with certain information (e.g. some extensions for the Super-BASIC language). Once programmed, you can only read information from it. If you expose a little window on it to strong ultra violet light (in a sealed container of course, you can buy special ones for this job) it will erase the program and you can then user a programmer device to save new information to it
EEPROM	Electrically Erasable Programmable Read-Only Memory, (see above) but programmed and erased electrically using field emission known as 'Fowler-Nordheim tunnelling' who first proposed the method
EPROM Slot	Term used to refer to a connection on the back of a QL which allows you to plug in a software ROM cartridge or small circuit board containing a ROM or EPROM
Emulator	A program which runs on one computer and allows that computer to run programs designed for a different system. For example, QL2K is a program which runs on a PC in Windows and runs QL software on the PC.
Endian	Refers to the layout of bits in a computer system, specifically whether the low end or high end bits and byte come first or last. A PC, for example, is Little Endian (which means little end of a value first), while processors such as the 68000 series are Big Endian (which means that the high value part of a number comes first).
Environment Variables	System which allows names to be defined and given values by one program (e.g. SuperBASIC) and which can be accessed by other programs. Provides a means of setting default values in a boot program for example, which can be detected by later programs. The Environment Variables software was originally supplied with the C68 C compiler software.
EOF	End of file. The EOF command is used to check if we have reached the end of a file yet.
Equivalent	As distinct from equals, this means that values are sufficiently close to being equal that to all intents and purposes they should be treated as equal. In QL programming, this is indicated by use of the double equals ("==") symbol to indicate that the values are "approximately equal". For strings, this means that lower and upper case are treated as being the same (e.g. IF a$="me" would match ME, Me, me and mE. For numbers, it is when values are so near to being equal to within several decimal places, according to Jan Jones in her book "QL SuperBASIC The Definitive Handbook", x==y will be true if (x-y) <= (y*1E-7), where (x-y) means the absolute or positive value of x-y
Error	When something in a program has gone wrong, the computer may tell you with an error value such as -15 or the message "bad parameter" when a value to something is not in the range expected.
Exception	A hardware condition triggered when something goes wrong in processing, for example an attempt to divide by zero, or the QL trying to access memory at a non-existent address.

Execute	Start a QL machine code program (as distinct from running a SuperBASIC program).
EXIT	Name of a keyword or act of leaving a loop structure such as a FOR or REPeat loop.
Expander	A plug-in circuit board for a QL which adds facilities such as a memory expansion, a floppy disk interface, hard disk interface or any combination of these.
Expansion Slot	Term which refers to a connection on the left hand side of the QL into which you can plug an expansion card such as a floppy disk interface or memory expansion device.
Extended Environment	See EE above.
FDD	Floppy Disk Drive
File Header	A short list of data about a file, which either precedes the main file itself, or may be held separately on a disk drive, which holds information such as the size of the file, date it was last updated, file type and so on.
FLP	Abbreviation for Floppy Disks. Most QL disk systems refer to disk drive number 1 as FLP1_ for example. FLP is an example of a directory device name.
Forward Slash	The "/" symbol on the keyboard, as distinct from the Back Slash symbol "\".
Frame Rate	A frame rate is the number of pictures shown per second and is also the time units used for the INKEY$ and PAUSE keywords. In most countries this corresponds to 1/50th second, while in the USA for example it might be 1/60th of a second.
FTP	File Transfer Protocol, a term for a method to transfer files via the World Wide Web
Function	A block of code, usually given a name, which is used to perform one or more specific tasks (usually calculations) within a program and return a result value. Think of it as a building block for a program. It can be called from other places in a program several times by different parts of a program, to avoid having to rewrite the same code over and over again.
Fuzz	Term used to describe one of the parameters of a QL BEEP sound command. The 'Fuzz' value (0-15) describes how "fuzzy", "blurred" or "distorted" the sound becomes when you use values of 8 to 15 in a BEEP command for the Fuzz parameter.
GAL	Gate Array Logic, a type of logic chip used in the Qubide, for example
GB	Gigabyte, for 1,024 Megabytes, used to indicate the capacities of modern very large hard disk drives. Some hard disk manufacturers tend to use Gigabyte to mean 1,000 Megabytes instead
GC	Gold Card, an expansion card for the QL giving a number of additional ports
GD2	Graphics Device Interface 2. The name for the system which allows more colours than the standard QL four or eight on the screen
GIF	Graphics Interchange Format, a graphics file format licensed from CompuServe
GLUE	A type of logic chip, as used on the QXL for example. Usually the chip which controls the interaction of other peripheral chips
GPL	General Public Licence, a type of licence published by the GNU project. It usually allows you to run a program, to study how a program works, modify it, give copies free to others, improve the program and release the improved version.
GPU	Graphics Processing Unit, a single chip normally dedicated to 3D graphic environments

A Serial Nightmare: The Story of the Ser-USB Drivers - Part 1

by Adrian Ives of Memory Lane Computing

I have a recurring nightmare.

I wake up in a cold sweat and discover that it is February 2010 again; and the drivers for the Ser-USB still have to be written.

The reason that this particular vision fills me with such dread is because now I know what is to come. Now I know how many thousands of hours it will take to tame the many incarnations of QDOS and to force thirty year-old serial ports barely fit to drive a dot matrix printer to accommodate the demands of a 21st century storage device. And if I had known then what I know now ...

... But this is a story that begins earlier.

It is October 28th, 2009. I had been in touch by e-mail with Tony Firshman concerning my non-functioning (then and now) ROMDisQ; somehow the discussion had turned to replacement storage devices, and I was considering whether a simple USB interface would be possible. It was then that Tony mentioned the USBWiz, a subject that he had broached previously on the QL Users List but which I hadn't picked up on. This was really the beginning.

Around that time Memory Lane Computing had been incorporated, formed in the wake of my taking voluntary redundancy. Although it hadn't been my intention to focus on creating new hardware, back then the idea of producing a workable USBWiz-based storage device for the QL was too interesting to resist.

It seemed straightforward. All that would be needed was an RS232 to TTL level converter (the USBWiz uses TTL serial levels) and a device driver. I remember the reasoning process that followed only too well: Let's see ... QUBIDE is public domain ... that has all the core directory device code that I need ... it just needs the hardware specific stuff removed and replaced with calls to an interface layer that emulates the IDE controller and generates USBWiz commands over the serial port.

How hard can that be? ...

This is the story of how the Ser-USB drivers were written. The highs and the lows (more of the latter, unfortunately) and a game of mental chess played out against the backdrop of multiple ROM versions, unreliable serial hardware, seemingly little interest from the QL community, and operating system design features that appeared purposely designed to frustrate the project.

This is also a story that almost didn't happen. There were several occasions when I decided that the obstacles were just too great and that the pursuit of a solution was not only a hopeless endeavour but a waste of time, effort and money.

Here at the end, in the final days of the Ser-USB project, when the 2.0 drivers are close to being publicly released, and other projects are imminent, I can look back and take stock. Yes, it probably was a waste of money. Viewed in simple profit and loss terms, Ser-USB is an investment that will never be recouped. On the other hand, the code libraries that have resulted from the Ser-USB project, the EDDE 2 universal driver core, the utilities and extensions; all of these are reusable. Already they are at the heart of the drivers for the QL-SD and the Q-BUS. The BDI driver for Q-emuLator uses the same core and work is advanced on EDDE 3 - a universal plug and play driver architecture that will allow new device drivers to be created by anyone with just a handful of hardware-specific routines.

So, from a USBWiz in a tatty black plastic project box with its overheating MAX232 chip mounted on veroboard, to custom PCBs and distinctive clear blue enclosures; from driver code that flipped into user mode to complete serial I/O, through the unwieldy Queue Manager, to the IOSS Retry Integration, Write As You Go map handling and the Extended Open Handler of the Ser-USB Legacy Driver; from an idea to a product ... this is a story that surely deserves to be told.

Part One: GENESIS

The first Ser-USB prototype was completed on the 19th of November, 2009, but it would be a further eight weeks before there was anything even approaching a device driver.

These pictures show how the original device was constructed inside a surplus black plastic project box, its level translator circuit and LED indicators constructed on veroboard.

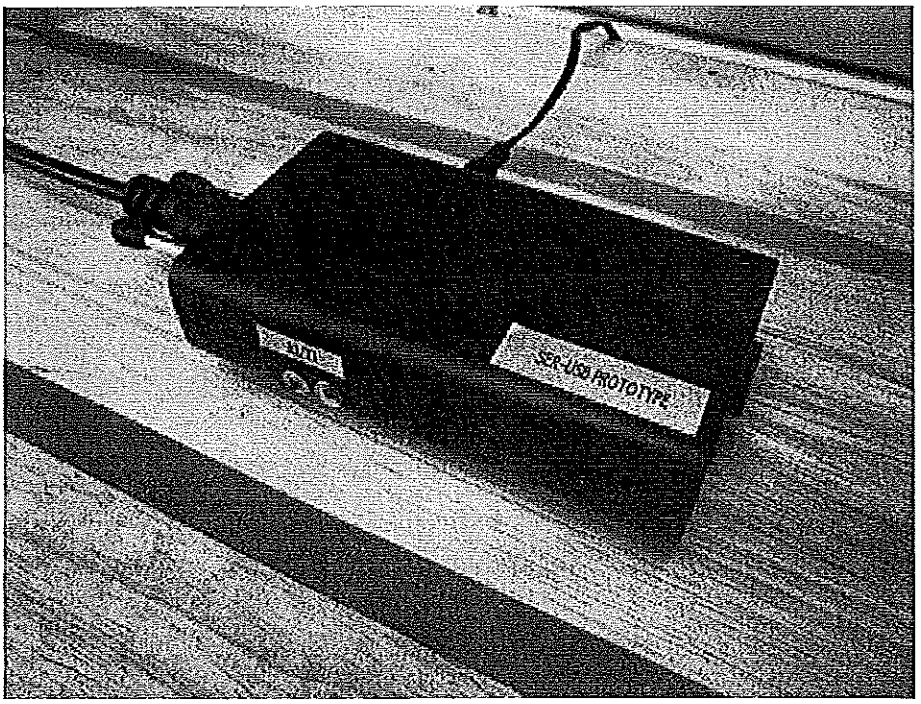


Illustration 1: The first Ser-USB prototype

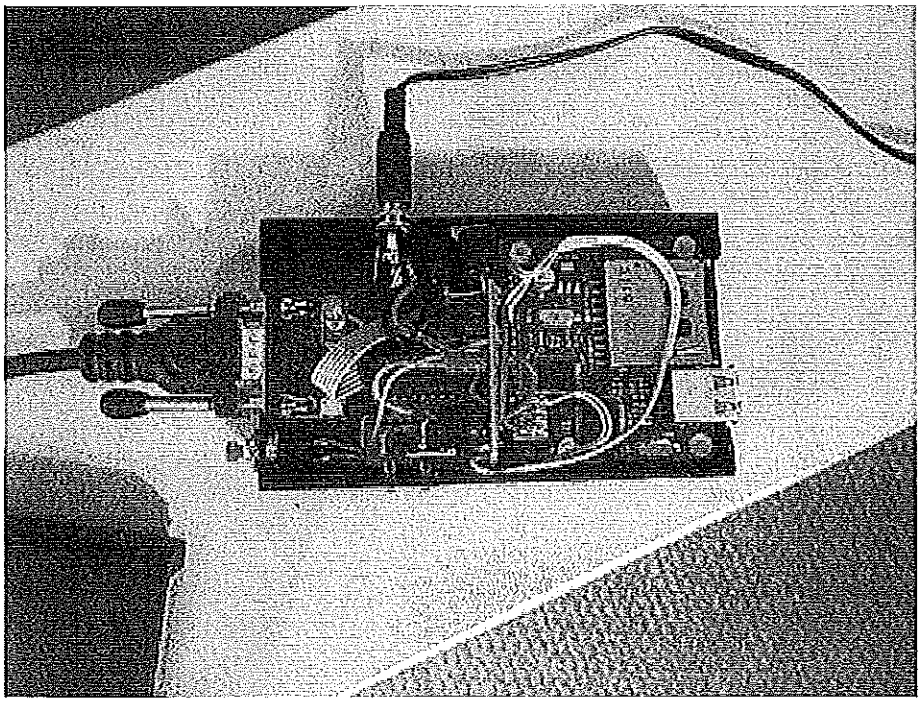


Illustration 2: Inside the Ser-USB prototype

The very first version of the Ser-USB driver is dated 11th January, 2010. It was version 0.01. It ran on an Aurora with a Super Gold Card, SMSQ 2.98, and a superHermes. It couldn't format anything, but it could copy files and display a directory. Very little else worked.

The work to create this version had started in November 2009 with the source code for the QUBIDE driver. I knew that the very first thing that had to be done was to deconstruct that driver, identify all of the hardware-specific code and replace it with code that spoke to the serial ports instead. It sounds simple when you say it quickly, but there was a lot more to it than that. I knew that it would be an impossible task to rewrite all of the hardware-specific code piece by piece so I took a different approach. I wrote an ATAPI emulator layer. A handful of routines that were called in place of the code accessing the IDE controller's registers that could then map those actions onto the USBWiz.

There was also the small problem of a disk drive architecture that was built around Cylinders, Heads and Sectors which had to be mapped to Logical Block Addresses (LBAs), but that could come later.

Writing the ATAPI emulator layer took most of December 2009 and continued into January 2010. More than half of that time was taken up simply by developing an understanding of what the QUBIDE code was actually doing. At that time all of the coding was done on a Windows PC with QPC2 and the Quanta QMAC assembler.

The hard work in v0.01 was done in a single module called usbdev. This contained the device driver initialisation code, a handful of control routines and the block I/O functions. It's interesting to read the XDEF list of that first iteration of the main USBWiz hardware-specific module: `init`, `wait_busy`, `reset_drive`, `read_params`, `usbwiz_rdsect`, `usbwiz_wrsect`, `drive_select`, `drive_link`, `drive_cmd`, `drive_capacity`, `drive_rns`, `drive_wns`.

Only three of these functions were to survive into the final production driver: `drive_select`, `drive_link`and `drive_capacity`. They are still in the code today, although they bear no resemblance to their distant ancestors. Perhaps the two most important functions, `usbwiz_rdsect`

and `usbwiz_wrsect` (read and write sectors) were destined to become `hw_read_lba` and `hw_write_lba` while `drive_rns` and `drive_wns` (Read Next Sector, Write Next Sector) ceased to exist when the ATAPI emulator layer was finally dispensed with.

Looking back over that code it is, without doubt, a mess. An untidy tangle of hacks and patches that somehow worked enough to create that very first version and provided me with the justification to go ahead with the project.

What I did not know at that time was that some serious problems were being disguised by developing under QPC2 and then running on the Aurora with SMSQ and superHermes. Most importantly, a tiny little problem concerning device drivers that call other device drivers which, in turn, are unable to complete transactions atomically. I have come to know this as the D3 Curse.

Register D3 is used to pass a timeout to all Trap #3 (I/O) calls. If it has a value other than zero it has a special significance; it tells the IOSS (Input Output Sub System) in QDOS that the trap should be retried until either (a) it returns 0 or an error code other than err.nc (Not Complete) or (b) the specified number of timer ticks in D3 have elapsed. -1 has the special significance of telling the scheduler to keep retrying forever until (a) is satisfied.

The problem is that if you have a slow device, such as a serial port, the serial driver may take hundreds of timer ticks to complete a trap #3 request, and it won't complete at all unless control is returned to the scheduler so that interrupts and other housekeeping can take place. But if you do specify a timeout in D3 when you make a trap #3 call from supervisor mode inside a device driver, the scheduler gets re-entered and this causes big problems! The recommended method is actually to use a zero timeout and to keep retrying the call in your own code with a loop. Unfortunately, if you do this with the standard QL serial port driver, the call will never complete.

And that was the problem at the heart of the Ser-USB driver. That was the dreaded ***D3 Curse!***

Even more Assembler Discussions

by George Gwilt and Norman Dunbar

Norman's *[ND]* answers to George's *[GG]* comments on Assembler - Part 29

[GG] In Part 29 Norman deals with what he calls a static menu inside an application window. SETW is used to produce the required window definition.

[GG] 1. My first comment is an apology for the fact that SETW does not allow a long enough name for Norman. A new version to allow the longest possible names to be entered is now available.
[ND] I shall be looking at the new version very soon. Thanks for a quick turnaround and fix.

[GG] 2. At the bottom of page 29 I think the number of objects (for the second information window) should be 0 and not 0.10.
[ND] Yes indeed. That should have been zero. I have actually asked Geoff to put up a correction, but that won't be necessary now as this will cover it.

[GG] 3. In step 14 Norman goes through the process of defining the colours for presentation of loose items. SETW allows a set of defaults here. The reason for that was that I got fed up having to type in all these colours, several of which would turn out to be not what I wanted. Perhaps SETW should have allowed these defaults to be configured.
[ND] That would possibly help when the users have defined their own colour scheme. I simply went through each step in order that we are all "on the same hymn book" as it were.

[GG] 4. On page 31, Norman suggests that the resultant _asm file should be saved somewhere. I agree with that. It is only too easy to switch off without saving the file. It was to prevent such annoyances that it was arranged that SETW can be configured to give directories for the saving of all three of the files which it produces. I find it a great comfort.
[ND] I always do my rough work on RAM1 or RAM2, so even with the configuration options correctly set, I'd still be there. It's something I've always tended to do since I obtained my first Trump Card all those years ago. Still doing it today. I used to forget to copy the file safely years ago, but I seem to be better at remembering now!

[GG] 5. I noticed with surprise on page 34 that the flag in the window definition wd0 was 258. This indicates a shadow size of 2 combined with the instruction to disallow the arrow keys moving the pointer. This means that Norman was using an old version of SETW. The correct value of the flag is given for version 7.06 or later. The value should be 2 and not 258. This error also applies to "clear flag" in appw0 on page 49. This should be 0 not 256.
[ND] Hmmm. I was sure I'd upgraded SETW. However, I'll be using the latest version from now on so it shouldn't be a problem in future.

[GG] 6. On page 35, in discussing how the menu objects are defined, Norman, quite rightly, indicates that they can be of any of four types, text, sprite, blob or pattern. I'm afraid that I thought it unlikely that anyone would really want anything but text in a set of menu items so SETW only allows these. This greatly simplifies the use of SETW in this area. I'm afraid also that SETW simply puts the text items where it can in the window provided. If there is not room for them all then scroll bars are automatically produced.
[ND] While the above may be true, the fact is, SETW makes the initial generation of the code to build menus etc quite easy. It is a matter of a few minutes to change things around to make different menu item types, of dimensions etc. SETW does the huge majority of the hard work.

[GG] 7. Norman shows how you can have indexes. Well, I await with enormous interest to see the program that actually produces these. I have tried, and failed, to have anything approaching an index appear on the screen using WM_INDEX. I even stepped through WM_INDEX using QMON and noticed the complete lack of index production. To find out whether indexes were produced by some other program in the system I searched the whole of the set of _asm files in SMSQ/E source code for the few items such as wwa_xind (which is the label in an application window attached to the pointer to the column index list). Answer came there none.
[ND] Yes. George and I have had an off-line discussion on the Index problem before Christmas. I was working on the fact that the QPTR manual was correct and accurate. George did what he does best and dived into the actual code to see what is working and what is not. George's findings show why there are no PE programs using indexes around. Simply put, they are not implemented.

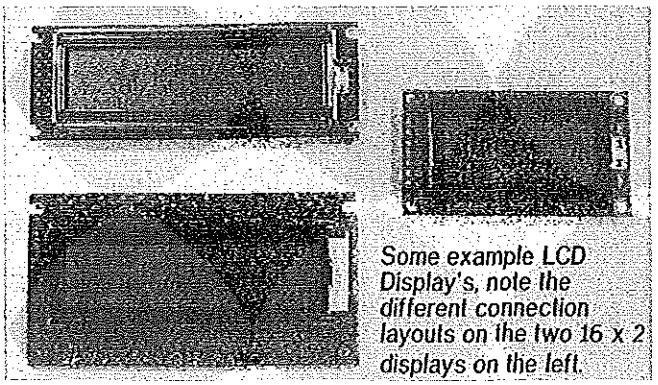
I2C Interface for QL Emulators – Part 4

by Ian Burkinshaw

In the first three parts of this series we have looked at the principles of the I2C bus and some of the devices that can be used. Such as parallel and AD/DA interfaces, RTC (Real Time Clock), RAM (Random Access Memory) and a digital potentiometer. This time we shall look at a practical use for the PCF8574 parallel device we looked at in part one.

In this article I am going to show you how you can drive a LCD (Liquid Crystal Display). In part one, I showed a circuit to be able to demonstrate input and outputs to and from the PCF8574 device. In this application we only need to use the output function of the PCF8574 since we are only going to drive the LCD display at a simple level. However the input/output capability of the PCF8574, can fully exploit all the available functions of the LCD displays such as these. See the HD44780.pdf in the references below, for further information on this.

LCD's (Liquid Crystal Display's) come in many different sizes, for example common ones are 8 characters by 2 lines, 16 Characters by 2 lines, 16 characters by 4 lines, 20 character by 1 line, 20 character by 2 lines and 40 character by 2 lines. They are easily available from the likes of Farnell and RS via mail order or via their websites. There are other suppliers as well.

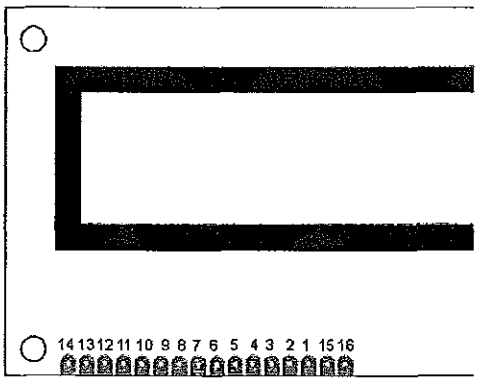


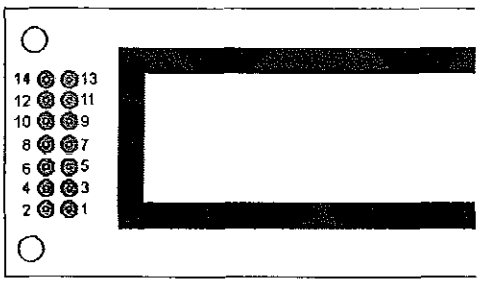
Some example LCD Display's, note the different connection layouts on the two 16 x 2 displays on the left.

One of the big advantages of these displays is that they have a standard interface and protocol using the Hitachi HD44780 driver chip. So any software you write will work with all versions of the displays. The only consideration is how many characters you wish to display at any one time, which should govern the size of display you use. LCD's come with or without back lighting - it is your choice which to use. However back lit versions do tend to be more expensive. Also check the display you are using has the HD44780 chip. Most do, but it is worth double checking.

Another word of caution, some modules require as low as -7V on their Vee (Contrast) pins, these versions will not work in this circuit - double check the datasheet for the device

The standard connection layouts of the displays are shown below. There are some variations out there, so again do check the datasheet for the LCD you are using. But these are the common ones.

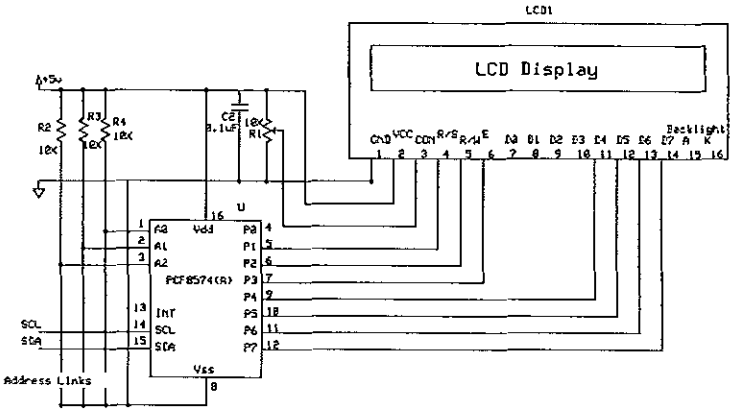




The pin out functions for all LCD types is shown below:

Pin no	Name	Function
1	Vss	Ground 0V
2	Vdd	+ve Supply (5V)
3	Vee	Contrast
4	RS	Register Select
5	R/W	Read/Write
6	E	Enable
7	D0	Data Bit 0
8	D1	Data Bit 1
9	D2	Data Bit 2
10	D3	Data Bit 3
11	D4	Data Bit 4
12	D5	Data Bit 5
13	D6	Data Bit 6
14	D7	Data Bit 7
15	BLA (If Fitted)	Back Light Anode
16	BLC (If Fitted)	Back Light Cathode

Below you will see the circuit diagram for the I2C to LCD project. As usual refer to part one which shows the connections to the BV4221 USB to I2C converter. The BV4221 provides the 5V supply, SCL, SDA and GND connections required. So the circuit is self powered from the BV4221 USB to I2C converter.



The circuit above is designed to exploit the full capabilities of the LCD module. However in this example we will only be sending data to the module, not reading any data from the module. So plenty of room for you to experiment.

The potentiometer R1 is the display contrast control, this can be a preset type, since once set, it does not need to be touched often. Note: with multi line displays, this may need to be adjusted if you change the number of lines to be displayed. So for example if is a two line display and you change it to one line operation from two line operation then the contrast will need adjusting. Best to start off with the pot turned the GND end of the travel to start with, then adjust for best contrast once you have run the program. You may see the top line of the display go dark at switch on, this is normal until the LCD display has been initialised, this will be explained later. Do not adjust until the display has been initialised and is displaying the characters you have sent it.

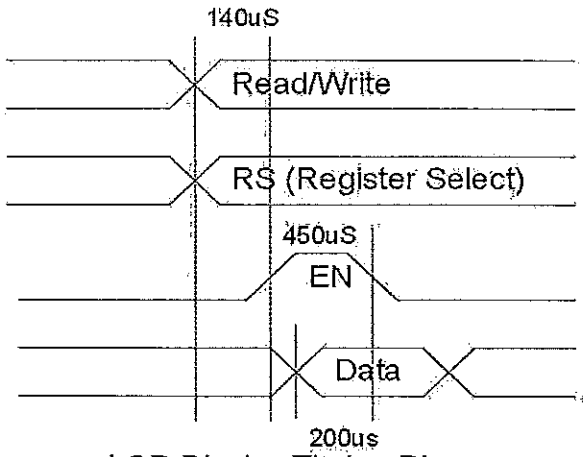
You will see from the above circuit diagram that we are using 7 bits from the PCF8574 to control and send data to the LCD module. These LCD modules have two modes of operation with

regards to data. The first being 8 bit words sent in one go. This requires all LCD data pins to be driven. That would then require two PCF8574, one for LCD control and a second to provide the data. Not the most efficient way. In the LCD's second mode of operation, we can use one PCF8574, the lower 4 bits being used (actually only 3 are used, there is a spare pin), for LCD control and the upper 4 bits for data. The data is now sent as two 4 bit nibbles, providing the 8 bit word the LCD requires.

The spare line from the PCF8574, could via a transistor (not shown) be used to control the back light of the LCD, if you use a LCD module fitted with one. Do not try to drive the back light directly from the PCF8574, since a fairly high current is required.

There is a process that your software needs to perform to get the LCD display working. First the LCD display needs to be initialised. See line 1120 (LCD_init) onwards in the listing below. The first process is to tell the LCD Display to work in 4 bit mode, which is command,'32'. This is sent as a single 8 bit word, since this is D5 bit =1 and the rest of the bits are 0, this is one of the reasons bit 0 to bit 4 has to be grounded so they really are 0. Next we have to set the LCD to accept 8 bits (as 2 nibbles, in this case) and that the display, in this case, is a two line one. The numbers sent are now split into two, the first nibble that is the last 4 bits then the second nibble which is the first 4 bits. All commands and data have to be sent this way from now on. The next command sent set the LCD display to, display ON, cursor ON and Cursor blinking. The last command to initialise the display is Entry mode, Cursor increment and No display shift. All the Commands are shown in the table below. After much experimenting and referring to the data sheets, the data is sent in a given sequence so at to replicate the timing diagram shown at the top of the next column.

Now the timing is not important, in fact the timings shown are minimum values as described in data sheets. However the order in which the control and data lines are delivered are important.



LCD Display Timing Diagram

So commands and data is sent as follows:

Command or Data	EN Line	RS Line
0	0	1
0	1	1
Data 1st Nibble	1	1
Data 1st Nibbel	0	1

Repeat for the second data nibble. The action of the EN line falling to zero(0), transfers the data within the LCD display.

The enable line is P3 from the PCF8574, so 8, (variable=en) is added to the data number to transfer the data in the LCD display. All the control codes are show in the table below.

Instruction	Instruction code										Description	Execution time (fosc=270KHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DDRAM address to 00H.	1.53ms
Return Home	0	0	0	0	0	0	0	0	1	-	Sets DDRAM address to 00H in AC and returns shifted display to its original position. The contents of DDRAM remain unchanged.	1.53ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	SH	Sets cursor move direction and enable the shift of entire display. These operations are performed during data write and read.	39μs
Display ON/ OFF Control	0	0	0	0	0	0	1	D	C	B	Set ON/OFF of entire display (D), cursor ON/OFF(C), and blinking of cursor position character(B).	39μs
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	-	-	Moves cursor and shifts display without changing DDRAM contents.	39μs
Function Set	0	0	0	0	1	DL	N	F	-	-	Sets interface data length (DL: 8-bit/4-bit), numbers of display line (N: 2-line/1-line), and display font type (F: 5x11dots/5x8dots)	39μs
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter.	39μs
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address Counter.	39μs
Read Busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.	0μs
Write data to CG or DD RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM).	43us
Read data from CG or DD RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM).	43us

Now the text can be sent to the LCD Display. This is done in the LCD_message procedure. The entire character set is also shown in the diagram to the right.

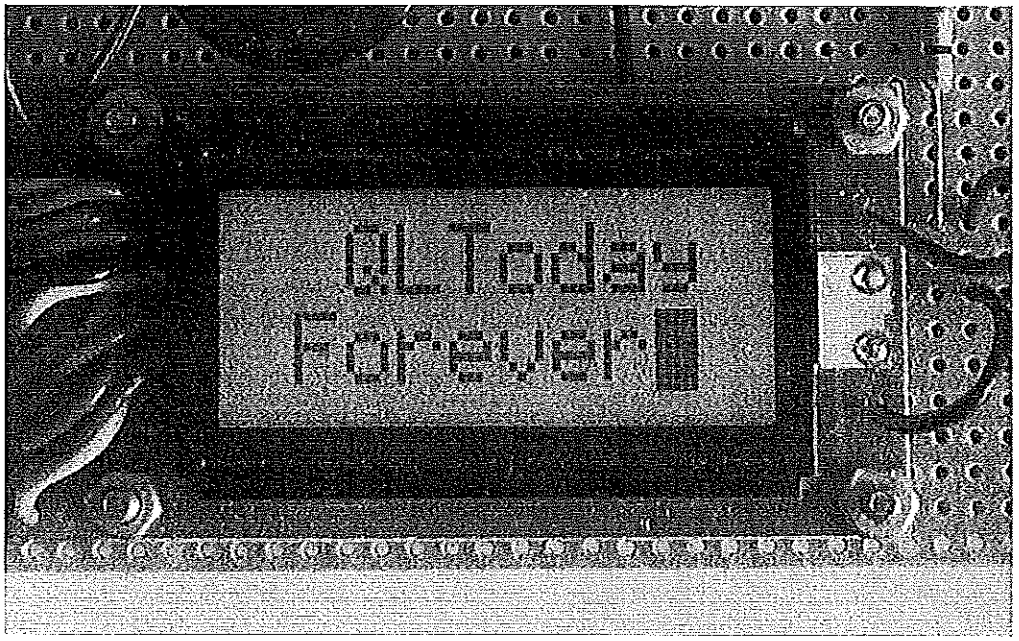
The following listing is just to demonstrate how to drive a LCD display in a basic manner. It has been deliberately slowed down so you can see the display respond to the commands. Clearly in normal operation you do not need to do this. The listing is fully annotated so you should be able to follow what is going on.

```
10 REMark LCD test routines
20 init
40 OPEN#3;ser2ir:REMark i=ignor hardware handshake, r=raw data
50 PRINT#3;CHR$(13);:REMark Carriage Return to set the baud rate in the USB to I2C converter,
   required on first pass to initialise USB to I2C converter.
60 print_reply:PRINT "Reply from sending CR."
70 PRINT
80 PRINT#3;"V";CHR$(13);:REMark Command to USB to I2C coverter for firmware version.
90 PRINT "Return USB Converter Version Number:-";
100 extract_read_data:PRINT d$:print_reply:REMark Prints version number reply from USB to I2C
    converter
110 PRINT
120 PRINT#3;"D";CHR$(13);:REMark Sets USB to I2C coverter to receive decimal numbers, default is
    hex numbers.
130 PRINT "Decimal Mode Selected"
140 print_reply:REMark returns a device address.
150 PRINT
160 :
170 init_LCD
180 LCD_message "QLToday"
190 move_second_line
200 LCD_message "Forever"
280 PRINT "End          ":CLOSE#3:STOP
290 :
1000 DEFine PROCedure init
1010 CLS
1020 BAUD 115200
1030 ram=174:REMark PCF8570 address, all address links open.
1040 parallel1=126:REMark PCF8574A address, all address links open
```

```
1050 parallel2=78:REMark PCF8574 address, all links open
1060 adda=158:REMark PCF8591 address, all address links open
1070 rtc=208:REMark DS1307 real time clock, one fixed address with this device.
1080 digpot=94:REMark DS1803 Digital Poteniometer, all link open
1090 DIM tdata(7)
1100 DIM days$(7,3)
1110 RESTORE
1120 FOR a=1 TO 7
1130 READ d$
1140 days$(a)=d$
1150 NEXT a
1160 DATA "Mon","Tue","Wed","Thu","Fri","Sat","Sun"
1170 END DEFine init
1180 :
1190 DEFine PROCedure print_reply
1200 c$=""
1210 REPeat loop
1220 a$=INKEY$(#3)
1230 b$=a$
1240 c$=c$&b$
1250 PRINT b$;
1260 IF a$=">" THEN EXIT loop
1270 END REPeat loop
1280 END DEFine print_reply
1300 :
1310 DEFine PROCedure non_print_reply
1320 c$=""
1330 REPeat loop
1340 a$=INKEY$(#3)
1350 b$=a$
1360 c$=c$&b$
1370 IF a$=">" THEN EXIT loop
1380 END REPeat loop
1390 END DEFine non_print_reply
1400 :
1410 DEFine PROCedure extract_read_data
1420 d$=""
1430 REPeat data_loop
1440 a$=INKEY$(#3)
1450 b$=a$
1460 d$=d$&b$
1470 IF a$=CHR$(10) THEN EXIT data_loop
1480 END REPeat data_loop
1490 END DEFine extract_read_data
1500 :
2000 DEFine PROCedure init_LCD
2010 rs=2:rw=4:en=8:REMark rs is register select, rw is read/write (only need for reading
     display rom, enable to transfer data to LCD display.
2020 PRINT#3;"s-";parallel1;" 0 p";CHR$(13):REMark set all outfrom from PCF8574 to low.
2030 non_print_reply
2040 load_LCD 32:REMark Sets LCD to 4 bit mode
2050 load_LCD 32:REMark Sets LCD to data length to 8 bit, 1st nibble
2060 load_LCD 128:REMark Sets LCD to data length to 8 bit, 2nd nibble
2070 load_LCD 0:REMark Sets LCD to Display ON, Cursor ON, Cursor Blicking, 1st nibble
2080 load_LCD 240:REMark Sets LCD to Display ON, Cursor ON, Cursor Blicking, 2nd nibble
2090 load_LCD 0:REMark Sets LCD to Entry Mode, Increment Cursor Position, No Display Shift, 1st nibble
2100 load_LCD 96:REMark Sets LCD to Entry Mode, Increment Cursor Position, No Display Shift, 2nd nibble
2190 END DEFine init_LCD:
2200 :
2210 DEFine PROCedure LCD_message(message$)
2220 mlen=LEN(message$)
2230 FOR mc=1 TO mlen
2240 ms$=message$(mc)
2250 ms=CODE(ms$)
2260 nib1=(INT(ms/16))
2270 nib2=ms-(nib1*16)
2280 nib1=(nib1*16)+rs
2290 nib2=(nib2*16)+rs
2300 load_LCD nib1
2310 load_LCD nib2
2360 PRINT ms$;" ";ms;" ";nib1;" ";nib2
```

```
2370 NEXT mc
2390 END DEFine LCD_message
2400 :
2410 DEFine PROCedure move_second_line
2420 load_LCD 192:REMark Move to start of second line 1st nibble
2430 load_LCD 0:REMark Move to start of second line 2nd nibble
2440 END DEFine move_second_line
2450 :
3000 DEFine PROCedure load_LCD(lcd_data)
3010 PRINT#3;"s-";parallel1;" ";lcd_data;" p";CHR$(13)
3020 PAUSE 4:non_print_reply:REMark delays required for the USB to I2C and display to respond
3030 PRINT#3;"s-";parallel1;" ";lcd_data+en;" p";CHR$(13)
3040 PAUSE 4:non_print_reply
3050 PRINT#3;"s-";parallel1;" ";lcd_data;" p";CHR$(13)
3060 PAUSE 4:non_print_reply
3070 PRINT lcd_data
3090 END DEFine load_LCD
3100 :
```

If all goes well you should get a display like this.



So now is the time for you to experiment with the initialisation as well as the characters. Also you can program the LCD Display with your own characters, read the reference documents on how to do this. There is lots of information on the web about these clever displays. So lots to have fun with, like scrolling displays. This article is just to get you going.

Next time we will take a break from the I2C interface and will look at using the parallel printer port on QXL, QPC2, Qemulator and Super Gold card QL's, as a simple output port and show more examples of how to drive LCD displays.

References

Farnell
http://uk.farnell.com/jsp/home/homepage.jsp

RS
http://uk.rs-online.com/web/

PCF8574(A) Data Sheet
http://www.nxp.com/documents/data_sheet/PCF8574.pdf
http://focus.ti.com/lit/ds/symlink/pcf8574.pdf

How to use intelligent LCD's Part 1
Everyday Practical Electronics, February 1997
Downloadable from http://www.wizard.org/auction_support/lcd1.pdf

How to use intelligent LCD's Part 2
Everyday Practical Electronics, March 1997
Downloadable from http://www.wizard.org/auction_support/lcd2.pdf

Character LCD Displays - Part 1
http://www.protostack.com/blog/2010/03/character-lcd-displays-part-1/

Hitachi LDC Driver chip, advanced stuff in here. But does show all the capabilities of LCD displays that use this chip.
http://www.sparkfun.com/datasheets/LCD/HD44780.pdf

Off Topic - Raspberry Pi Tinkerers Rejoice!

by Norman Dunbar

As QL users, even though the platform has effectively been 'dead' for many years, we are mostly tinkerers by nature. QUANTA, the long lived and much loved user group, stands for QL Users And Tinkerers Association, so, the Raspberry Pi should appeal to quite a few of us.

Simon Balderson/Bryan Horstmann were the first to bring this new device to the QL Users list back in June 2011 which was not the first I'd heard of it as I'd seen the video of the device on the BBC's very own Rory Cellan-Jones' blog at **http://www.bbc.co.uk/blogs/thereporters/rorycellanjones/2011/05/a_15_computer_to_inspire_young.html.**

The device has been designed and built by a number of people including one David Braben - the same David Braben who created Elite, way back in the eighties, for the BBC Micro.

The Raspberry Pi Foundation

The Raspberry Pi Foundation is a charity whose aim is to get kids programming again and came about after it was discovered that students starting courses in computing at university couldn't actually program. IT "education" these days consists of learning how to use Microsoft Office (other - and better - office suites are available of course) which is fine if you want to be a typist or secretary, but of little use in Computer Science for example.

On the main website, the About Us page details the background of the Raspberry Pi and how it came to be. The following is not a direct copy of that page, but is a brief introduction to the Foundation and the people involved. If you wish to read more, head over to the website. (See below for links and URLs.)

Eben Upton is a hardware guru and basically started it all when he was still in Academia. Applications for the Computer Science courses at the University he worked at, were dropping off and the skill sets of the applicants was falling. He built a number of prototypes over the years - some of which you can see on the web site.

Around 2008, Eben and a group of like minded friends and colleagues got together and started the Foundation. This group of people became the trustees of the charity.

David Braben is a "rock star" games designer and has a contact book "as long as your arm". He has been responsible for much of the publicity the Foundation has received and for raising sponsorship deals to help the charity.

Jack Lang, a Cambridge academic and business angel - who worked on the original BBC micro all those years ago - runs the business side of things and is the guardian of a warehouse full of components ready to be assembled into Raspberry Pi boards.

Pete Lomas is the MD of a hardware design and manufacturing company. He has been responsible for designing and building the early boards and for minor bug-fixes to the current beta boards on Ebay. He also has a natty line in tartan trousers!

Professor Alan Mycroft and Dr Rob Mullins, both from the Cambridge University Computer lab, have provided much of the educational direction for the project.

Liz Upton, wife of Eben, is not a trustee, but volunteers for many tasks such as photography, making the videos you see on the web site, moderating the forum and generally answering the same questions over and over from people who don't read the FAQ.

Specifications

The Raspberry Pi comes (or will, early in 2012 it is hoped) in two flavours - much like the BBC Micro from the eighties - the Model A and the Model B.

The Model A is expected to cost around $25 or roughly £16 while the Model B is intended to be around $35 or £22 - cheap enough to simply replace if you break it. The specifications of the two models are:

The Pi is 85.60mm by 53.98mm by 17mm, with a little overlap for the SD card and connectors which project over the edges. It weighs 45g. In other words almost identically sized to a credit card. There is not a case - you are expected, if you wish, to supply/build your own. It is hoped to supply cases at a later date.

Power is from an easily available 5V micro-USB power supply much loved of mobile phone manufacturers and most households should have at least one lying around! Power can also be supplied over the GPIO pins on the board. Power supplies are *not* supplied in the box with the Pi. It is predicted that the Pi will also run off 4 AA batteries.

Power usage is 2.5 W for the Model A and 3.5 for the Model B - as tested on the alpha boards. The final specification and power usage may go up as well as down. (As they say!)

Networking is via a 10/100 wired Ethernet on the Model B only. The Model A has no on board networking. In both cases WiFi networking will be available via a USB Dongle - which you will supply yourself.

The Model A has one USB connector while the Model B has two. You may have to supply a small USB hub to get all your devices connected.

The processor is an ARM11 device. This means that it will not run Windows programs compiled for Intel processors. The CPU runs at 700MHz - which doesn't sound much, but watch it in action on the web site before complaining! The whole lot is actually a System On a Chip (SOC) and is all wrapped up in a single Broadcom BCM2835 chip consisting of the CPU, GPU, DSP and the SDRAM.

The SDRAM memory is 128 Mb on the Model A and 256 on the Model B.

ArchLinux, Debian and Fedora Linux will be supported from day one. It is also hoped to see support from other Linux distros later. In addition, there might be a RiscOS release for the Pi. The OS will be supplied on an SD card and other cards preloaded with different OSs will also be on sale. It was originally intended that Ubuntu would also be available but because of issues with newer releases of Ubuntu and the ARM processor Ubuntu might not be available initially. Anything compiled for the ARMv6 CPU should work.

Sound is output via the HDMI connector to your TV or via a dedicated 3.5mm stereo socket on the main board. There is no sound input, but as ever, a USB device such as a microphone will do the necessary for you.

Video is available on the HDMI or composite outputs. VGA is not available though, although it is said that DVI-D will be.

HD output is the full 1080p standard.

There is no keyboard and no mouse supplied. You are required to connect your own via the USB interface. You can buy a decent quality, cheap keyboard and mouse from Asda or a similar outlet.

Expansion boards are in the pipeline. The first is the "Gertboard" names after Gert, one of the designers and the second is a Real Time Clock board - the Pi doesn't have an RTC built in.

So What Is It?

The Raspberry Pi is a small computer system that is hoped, will interest kids in starting to actually write their own programs again - like we all did back in the early eighties with our Spectrums, and QLs. Some people had Commodores as well, I'm told, but they have recovered now! It will look remarkably similar to this:

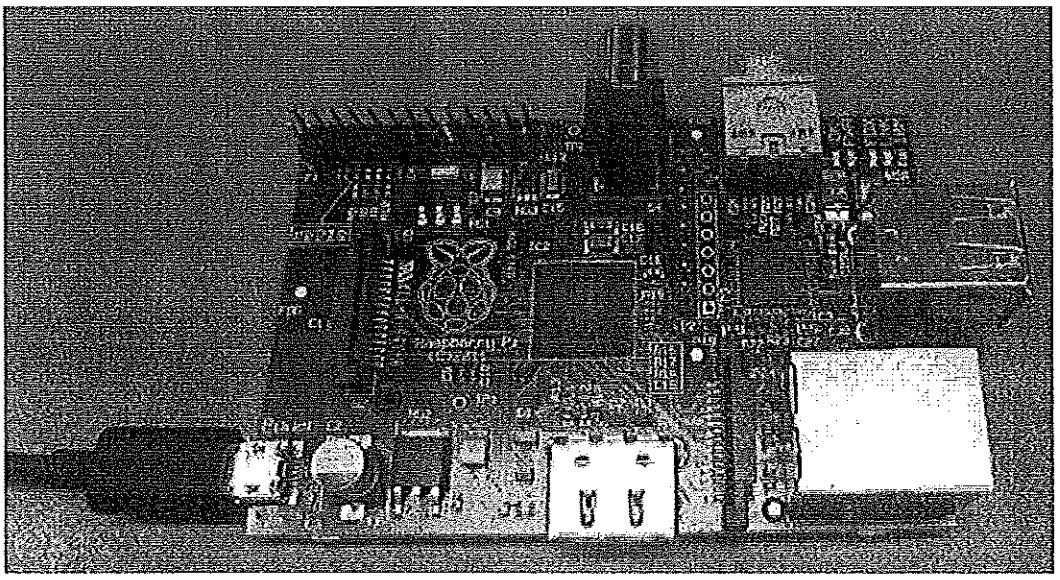


Image courtesy of the Raspberry Pi Foundation. Used with permission.

Kids nowadays don't know how to program, ICT at school is nothing more that learning to use Word or Excel (Your schooldays are "sponsored" by Microsoft - go figure!) and industry needs people who can program, not typists. Granted, typing skills are a good thing - I still use three

JOCHEN MERZ SOFTWARE

Kaiser-Wilhelm-Str. 302
47169 Duisburg, Germany
Fax +49 203 502012
EMail: SMSQ@J-M-S.com

QPC2 Version 3 + SMSQ/E Software QL-Emulator for PC's			**EUR 59,90**
QPC2 Version 3 - Upgrade from QPC2 Version 2			**EUR 19,90**
QPC2 Version 3 - Upgrade from QPC2 Version 1			**EUR 39,90**
QPC Print - printer emulation driver for QPC			**EUR 39,90**
BUNDLE: QPC2 and QPCPrint		**ONLY**	**EUR 79,90**
Agenda Agenda program for WMAN and Prowess	[V1.09]		**EUR 14,90**
Suqcess Database front-end for WMAN	[V2.05]		**EUR 19,90**
QD2003 Pointer-Environment-Editor	[V8.01]		**EUR 29,90**
QD2003 Upgrade from Version 9 and older	[V8.01]		**EUR 14,90**
QMAKE Pointer-driven MAKE for GST/Quanta Assembler	[V4.31]		**EUR 14,90**
BASIC Linker	[V1.21]		**EUR 14,90**
WINED Floppy/Harddisk Sector- & File-Editor	[V1.26]		**EUR 14,90**
FiFi II File-Finder - Extremely useful!	[V4.31]		**EUR 14,90**
FiFi II Upgrade from Fifi Version 3 or older	[V4.31]		**EUR 9,90**
EPROM Manager	[V3.02]		**EUR 14,90**
QSpread2003 Spreadsheet Program	[V4.04]		**EUR 29,90**
QSpread2003 Upgrade from Version 3 and older	[V4.04]		**EUR 14,90**
QPAC I Utility programs	[V1.11]		**EUR 19,90**
QPAC II Files, Jobs & other Things	[V1.45]		**EUR 29,90**
QTYP II Spell checker	[V2.17]		**EUR 19,90**
QPTR Pointer Toolkit	[V0.30]		**EUR 29,90**
DISA Interactive Disassembler	[V3.04]		**EUR 29,90**
CueShell	[V2.14]		**EUR 29,90**
CueShell for QPC	[V2.14]		**EUR 14,90**
SER Mouse software mouse driver for serial mice			**EUR 10,00**
EasyPTR Version 4	[V4]		**EUR 59,90**
EasyPTR Version 4 - Upgrade from earlier versions	[V4]		**EUR 39,90**
QDT - QL Desktop program			**EUR 59,90**
QMENU Version 8 - with new, printed Manual	[V8.02]		**EUR 24,90**
QMENU Version 8 - Update from earlier Versions, also with printed manual			**EUR 17,90**
QMENU Version 8 - New/Update for QL Today subscribers, with prtd manual		**ONLY**	**EUR 14,90**

Please add EUR 4,90 for postage to all destinations - Germany, Europe, Worldwide!

We accept VISA, MasterCard & Diners Club online and offline! Details for money transfers:

- Deutschland: Jochen Merz, Account 493 50 431, Postbank Essen, BLZ 360 100 43
- Österreich: Jochen Merz, Account 85055317, PSK Wien, BLZ 60000
- Switzerland: Jochen Merz, Account 60-690080-4, PostFinance, Clearing-Nr. 09000
- The Netherlands: Jochen Merz, Gironummer 3258439, Postbank NL Amsterdam
- and from all other countries in EUR with IBAN and BIC to account Jochen Merz, Deutsche Postbank AG, IBAN: DE21 3601 0043 0611 1004 37 / BIC: PBNKDEFF 360
- UK customers can pay in £ (convert EUR prices above to £ by multiplying with 0.89) to Jochen Merz, Account 83795395, Citibank UK, Sort code 30-00-45 or send cheques in £ - no fee for UK sterling cheques!
- US customers can pay in US$ (convert EUR prices above to US$ by multiplying with 1.36) - no fee for US cheques in US$!
- If you wish to pay via paypal, send money to Paypal@J-M-S.com

Cheques payable to Jochen Merz only!

Price list valid until 30th of April 2012

IF YOU WISH TO BUY ANY OF THE ABOVE ITEMS AT THE VIENNA MEETING, PLEASE LET ME KNOW IN ADVANCE!

fingers after 30 odd years in IT - but programming skills are desperately short of supply in the UK.

This lack of programing skills was noticed and the Raspberry Pi Foundation was set up in an effort to try and reverse the trend. It must be popular, the current government have jumped on the bandwagon as well.

Remember the days when you plugged in your home computer, waited for it to boot up, and then sat there looking at a prompt that said "Ready" or similar? Those days are back again. The Raspberry Pi will give you the ability to start doing something straight away without needing to pay huge amounts of money for a development system.

On the Forum, there is many a discussion about the ideal language for beginners - BASIC takes a hammering as being "unsuitable" for Beginners. I think they are wrong - and I've told them, but each of us has our own opinions. Python - not a language I would teach beginners - seems to be quite popular and is, to its credit, as "instant" as BASIC in getting things done quickly.

It would be nice to get SuperBasic running on a Pi - that's a project for another day - bearing in mind that the Original S*Basic was written in pure 68000 Assembler and that simply will not run on an ARM chip - a complete rewrite would be required.

The whole purpose of the Raspberry Pi is to get us all back into the habit of writing our own programs to do "stuff" instead of buying something that does it for us. This is right up the alleyway of all us tinkerers.

I have to admit that I'll be buying at least one of these devices when they go on general sale, at least one! However, what I'm intending to do with it when I do get one is a complete unknown at the moment. I don't care, I just want to play, and learn - just like the old days again.

Currently there are quite a few videos on the web site showing various features of the Pi. The 1080P full HD can be seen in action as well as various other useful demos of various applications (Quake) having been ported or simply just running on a bare bones Pi connected to a TV. It's impressive.

Equally impressive is just how popular this project has become. The web site is busy as are the forums and lots of people keep asking the age old questions "where do I buy one?".

Unlike Sinclair of old, no money will change hands until devices are physically available and ready. The Foundation doesn't want or need to get into a position of being accused of taking money to carry out beta work etc. They are up front and extremely honest - they don't want any money from you until you can actually buy something.

They do have an online shop though - go to the main web page and click on the "shop" link. It was set up to test out the purchasing processes and at present sells only a Raspberry Pi sticker for £1.10 including postage and packing. They sold out of the first batch extremely quickly and had to order more it - such was the interest.

As with many things popular, once the idea takes off, the hangers on arrive. A web site in Russia was advertising Raspberry Pis for sale even when the Alpha boards hadn't been available - but that web site was soon shut down. There are no other web sites able to sell the Pi, so don't be tempted elsewhere.

Unless, of course, you wish to be in at the very beginning. Ebay is selling the first 10 boards produced. Each is numbered and they are proving extremely popular. Board number 1 is currently being bid for at the cost of £3,300 - yes, that's not a misprint, £3,300 for a £22 computer!

Others in the series range from £780 to over £1,000 each! All proceeds from these 10 boards will go to the charity to help supply Raspberry Pis for schools. Stunning.

When the boards go on general sale though, you won't have to pay nearly as much - £16 or £22 should cover the bare board - you'll need a keyboard, mouse and a TV or similar and I'd advise adding a USB hub to multiply the ability to connect things.

The Gertboard

As I said above, Tinkerers rejoice! One of the developers - Gert van Loo - has designed the Gertboard to allow your Raspberry Pi to be interfaced with the world at large. See the links below for the place to go for an introduction to what looks to be a stunning add on for the Pi.

At the moment it appears that this board will be available only as a bare board and a list of parts you need to buy, however, there is talk of maybe perhaps a full or part kit being available some time.

The board has lots of flashing LEDs, motor controller, an PIC for Analogue to Digital conversions and so on and on and on! I want one!

Useful Links

The following is a list of useful links regarding the Raspberry Pi.

http://www.raspberrypi.org/ - the main website. You can also join the mailing list here - but be aware that only "big" announcements use the mailing list, that way, you don't feel as if you are being spammed. Check the main website for all the latest news.

http://www.raspberrypi.org/forum - the forum. Lots of discussions, questions and answers here. Join up and join in. (Try not to get banned though!)

http://elinux.org/RaspberryPiBoard - the wiki. Here you'll find full specs etc and lots of useful information.

http://elinux.org/RaspberryPiBoard#News_articles_and_blog_posts_about_Raspberry_Pi - a Wiki page with numerous links to articles etc about the Raspberry Pi.

http://www.raspberrypi.org/archives/411 - Gert van Loo's add on "gertboard" for the Raspberry Pi.

http://www.raspberrypi.org/forum/educational-applications/gertboard - more details of the gertboard.

The Next Volume

Thanks to your continuing support, we are now looking towards volume 17. The aim of 32 pages per issue is more than fulfilled, and the DVD has been a great success too.

The early renewals have been a great help and cost reduction, which allowed us to add many more pages than planned to the individual issues.

We would like to continue with this process and ask you to renew as early as possible. We take the risk and keep the subscription rate at the same price as for the current issue (although printing costs have gone up as of last issue). There is also the risk of the postage being raised during the volume, but I hope it won't happen or - at least - be only 2 or 3%. Let's all be positive about it, it will work out somehow, I hope.

We are even thinking about adding another extra goodie to one of the forthcoming issues. One possibility would be a CD, containing all the issues of volume 16 and other stuff. Rainer has already offered to scan the volumes.

But we are also thinking about other things ... if you come up with an idea, please write to us and see if we can turn it into reality. Please remember, budget is tight and weight means high shipping costs.

Please renew by using one of the many methods - as early as possible!

The Next Issue

We plan to have the next issue ready for you at the Vienna meeting - which is ealier than the standard schedule of the 4th issue of a Volume. As always, it depends on how quickly we get reviews, articles etc. So please send it as early as possible!

We need more material, as always. The more we get and the sooner we get it, the quicker the next issue will be in your hands, and the better it will be. Hope to meet you at one of the forthcoming QL shows - your QL Today Team!

The QL Show Agenda

QUANTA AGM 24./25. of March 2012

Quanta's 2012 Workshop and AGM will be held on 24-25th March 2012 at the 3rd Davyhulme Scout Headquarters, "The Endeavour", Conway Road (off Lostock Road), Davyhulme, Manchester M41 0TF. The venue is very easy to find - come off the M60 motorway at Junction 9 and take the West exit, signposted 'B5158 Urmston and Davyhulme'. Take the second right turning off this road (Lostock Road) onto Conway Road. The venue is down this lane on the right, near the first corner of the lane. If you need a map, use one of the mapping services on the Internet (e.g. Google maps or Streetmaps) to look for postcode M41 0TF.

All are welcome to this meeting, members and non-members alike. Whether you are a QL enthusiast or just curious about Sinclair's little multi-tasking black box. Admission is free. Tables will be available for those who would like to bring along their computer.

The Saturday workshop opens at 11AM on the Saturday (doors open at 10AM for setting up). On Sunday, the meeting opens at 10AM and the Annual General Meeting will be held at 2PM. Meet the experts! Come along and ask questions and discuss your systems with traders and committee members. So far, we expect RWAP Software and Memory Lane Computing to be there, so if you'd like to see the Ser-Usb device for the QL, or discuss Peter Graf's MDV slot SD card system, come along and meet Adrian Ives of Memory Lane Computing, for example. George Gwilt also hopes to be present to give a talk.

For those coming from a little further afield, please contact Quanta Secretary or Chairman for help with finding place to stay in the area or for public transport details. For those coming from far enough to fly in, Manchester airport is just a short distance away.

Homepage: www.quanta.org.uk

Vienna 7./8./9./10. of June 2012

The meeting at Prottes (near Vienna) is fixed, according to organizer Gerhard Plavec.

The time to reserve in your agenda is the 7th to 10th of June 2012. It is a "long" weekend, like it was last year: Thursday is a bank-holiday (at least in Germany and Austria), so it should make the visit easier.

The schedule for the days is similar to 2010, but other sightseeing highlights are planned. The estimated schedule printed in the previous issue has slightly changed. (The planned visit of the tramway museum moved to Sunday).

Gerhard has updated the website, so please visit www.kuel.org for details!

J-M-S plans to be at the meeting on at least two days, like last year. As Saturday will be the main day (with barbecue etc.), this will most likely be one of the days (if weather permits, of course). If you want me to bring anything you'd like to purchase (like QL Today back-issues), then please contact me (Jochen) a week in advance so that I can prepare it for you!

Hope to see many of you there - looking forward to another great meeting!